



UNIVERSIDADE DO VALE DO TAQUARI - UNIVATES

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

CURSO DE ENGENHARIA DA COMPUTAÇÃO

**GERENCIAMENTO CENTRALIZADO DE DISPOSITIVOS  
DISTRIBUÍDOS DA IOT**

Roberto Santin

Lajeado, novembro de 2017

ROBERTO SANTIN

## **GERENCIAMENTO CENTRALIZADO DE DISPOSITIVOS DISTRIBUÍDOS DA IOT**

Trabalho de conclusão de curso apresentado na disciplina de Trabalho de Conclusão de Curso, do Curso de Engenharia da Computação, da Universidade do Vale do Taquari - Univates, como parte da exigência para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Me. Alexandre Stürmer Wolf

Lajeado, novembro de 2017

# **GERENCIAMENTO CENTRALIZADO DE DISPOSITIVOS DISTRIBUÍDOS DA IOT**

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia da Computação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador:

---

Prof. Me. Alexandre Stürmer Wolf, UNIVATES  
Mestre pela PUC – Rio de Janeiro, Brasil

Banca Examinadora:

Prof. Ms. Alexandre Stürmer Wolf, UNIVATES  
Mestre pela PUC-Rio – Rio de Janeiro, Brasil

Prof. Ms. Juliano Dertzbacher, UNIVATES  
Mestre pela UFRGS – Porto Alegre, Brasil

Prof. Ms. Luis Antônio Schneiders, UNIVATES  
Mestre pela UFRGS – Porto Alegre, Brasil

Coordenador do Curso de Engenharia da  
Computação:

---

Prof. Dr. Marcelo de Gomensoro Malheiros

Lajeado, novembro de 2017

Dedico este trabalho aos meus pais,  
Moacir A. Santin e Irene D. D. Santin, que  
sempre me apoiaram em todos os momentos do bacharelado.

## **AGRADECIMENTOS**

Aos professores, em especial Alexandre Stürmer Wolf, Marcelo Malheiros, Fabrício Pretto, Luis Antônio Schneiders, que sempre apoiaram todas as minhas decisões durante o bacharelado, colaborando diretamente para o desenvolvimento deste trabalho.

Aos meus irmãos Luciane Santin, Ronaldo Santin (*in memoriam*), Ricardo Santin, que sempre estiveram presentes na minha vida pessoal.

À minha namorada Luiza, por estar sempre apoiando as minhas ideias e por ter compartilhado dessa trajetória junto a mim com total compreensão e companheirismo.

Aos Colegas Mateus Silva, Matheus Netto, Mateus Marmitt, Denner Erthal, Ivan Lampert, Maiquel Ludwing, Luis Henrique Galimberti, Willian Valer, Rodolfo Prediger Helfenstein, pela troca de experiências, auxílios e por todos os momentos compartilhados durante a formação acadêmica.

Aos amigos Eduardo Lisott e Rodrigo Lanzini por terem compreendido a minha ausência durante os churrascos para concluir a formação acadêmica.

## RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação de software para controle de dispositivos da Internet das Coisas - *Internet of Things* (IoT). A aplicação permite a integração de dispositivos com diferentes protocolos e plataformas, através da coleta e armazenamento de informações relevantes e envio de configurações para os dispositivos. A aplicação centralizadora, dispõe de um formulário dinâmico, desenvolvido para integrar a comunicação, permitindo enviar funções e receber dados do dispositivo gerenciado. Para alcançar os objetivos propostos, são utilizados protocolos de comunicação de dados, exploração de bibliotecas que agilizam o desenvolvimento de software, aliados a ferramentas e métodos já explorados pela literatura e profissionais da área. O protótipo desenvolvido utiliza a plataforma de hardware da Raspberry Pi 3, porém pode ser utilizado em dispositivos com características semelhantes de conectividade, memória e processamento. Para validar estes protótipos desenvolvidos, foi realizado um experimento prático, monitorando e controlando a temperatura de um *datacenter*. O experimento prático e a prova de conceito demonstraram a utilidade de unificar o controle das informações coletadas dos dispositivos distribuídos na aplicação centralizadora, permitindo realizar averiguações e monitoramento constante.

**Palavras-chave:** Internet das coisas, Protocolos de Comunicação, Gerenciamento de Dispositivos, IoT.

## **ABSTRACT**

This work presents the development of a software application to control Internet devices of Things - Internet of Things (IoT). The application allows the integration of devices with different protocols and platforms, by collecting and storing relevant information and sending configurations to the devices. The centralizing application has a dynamic form, designed to integrate communication, allowing you to send functions and receive data from the managed device. To achieve the proposed objectives, data communication protocols, library explorations that streamline software development are used, together with tools and methods already explored by the literature and professionals in the field. The developed prototype uses the Raspberry Pi 3 hardware platform, but can be used on devices with similar connectivity, memory and processing features. To validate these developed prototypes, a practical experiment will be carried out, monitoring and controlling the temperature of a datacenter. The practical experiment and the proof of concept demonstrate the usefulness of unifying the control of the information collected from the distributed devices in the centralizing application, allowing to make inquiries and constant monitoring.

Keywords: Internet of Things, Communication Protocols, Device Management, IoT.

## LISTA DE FIGURAS

Figura 1 – Raspberry Pi 3.....	13
Figura 2 – Arduino Industrial 101 .....	14
Figura 3 – Beaglebone Black .....	16
Figura 4 – Node MCU ESP-12E.....	18
Figura 5 – Etapa de compilação.....	22
Figura 6 – A Figura (a) apresenta uma Rede sem fio com uma estação base. A Figura (b) apresenta uma Rede tipo Ad hoc.....	26
Figura 7 – Canais e Frequências 802.11b .....	27
Figura 8 – Protocolo MQTT .....	34
Figura 9 – Arquitetura do sistema, Barboza (2015).....	36
Figura 10 – Arquitetura do sistema, Silva (2014) .....	37
Figura 11 – Arquitetura do Sistema.....	44
Figura 12 – Arquitetura de rede do Protótipo .....	45
Figura 13 – Diagrama de Caso de Uso .....	46
Figura 14 – Diagrama de Sequência.....	46
Figura 15 – Interface de acesso a aplicação centralizadora .....	47
Figura 16 – Menu de opções.....	48
Figura 17 – Formulário dinâmico.....	49
Figura 18 – Histórico de Eventos .....	50
Figura 19 – Cadastro de Usuário .....	51
Figura 20 – Cadastro de Dispositivos.....	52
Figura 21 – Modelo relacional do banco de dados.....	53
Figura 22 – Diagrama ligação diodo emissor de luz e sensor BME 280 .....	55
Figura 23 – Processo de troca de mensagens .....	57
Figura 24 – Retorno das propriedades e atributos .....	59
Figura 25 – Emissão do Certificado auto assinado .....	62



Figura 26 – Certificado auto assinado.....	63
Figura 27 – Histórico de eventos recebidos no monitoramento do Datacenter .....	65

## LISTA DE CÓDIGOS

Código 2 – Código exemplo ativando porta de saída 01.....	23
Código 3 – Código exemplo de uma requisição GET.....	30
Código 4 – Retorno do servidor.....	31
Código 5 – Estrutura dos atributos e valores JSON.....	32
Código 6 – Realizando uma conexão com o broker.....	34
Código 7 – Realizando uma publicação de temperatura para o broker.....	35
Código 8 – Acionamento do diodo emissor de luz .....	56
Código 9 – Captura de dados do Sensor BME 280.....	56
Código 10 – Exemplo da sincronização com o protocolo.....	58
Código 11 – Formulário dinâmico.....	60
Código 12 – Configuração Apache Tomcat.....	63

## **LISTA DE TABELAS**

Tabela 1 – Especificações das plataformas apresentadas .....	19
Tabela 2 – Padrões IEEE com seus principais aspectos .....	29
Tabela 3 – Métodos de solicitação HTTPS .....	31
Tabela 4 – Comparativo entre os trabalhos analisados .....	38

## LISTA DE ABREVIATURAS

AES	<i>Advanced Encryption System</i>
API	<i>Application Programming Interface</i>
ARM	<i>Advanced RISC Machine</i>
CMD	<i>Command Prompt</i>
CoAP	<i>Constrained Application Protocol</i>
CPU	<i>Central Processing Unit</i>
CSI	<i>Serial Interface</i>
CSS	<i>Cascading Style Sheets</i>
DSI	<i>Display Serial Interface</i>
GHz	<i>Giga Hertz</i>
GPIO	<i>General Purpose Input/Output</i>
HDMI	<i>High Definition Multimedia Interface</i>
HTML	<i>HyperText Markup Language</i>
HTTP	<i>Hypertext Transfer Protocol</i>

IDE	<i>Integrated Development Environment</i>
I/O	<i>Input / Output</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
JDK	<i>Java Development Kit</i>
JSON	<i>JavaScript Object Notation</i>
JSP	<i>JavaServer Pages</i>
JVM	<i>Java Virtual Machine</i>
LED	<i>Light Emitting Diode</i>
LPDDR	<i>Low Power Double Data Rate</i>
MHz	<i>Mega Hertz</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
M2M	<i>Machine to Machine</i>
OSI	<i>Open System Interconnection</i>
PWM	<i>Pulse Width Modulation</i>
RAM	<i>Random-Access Memory</i>
RISC	<i>Reduced Instruction Set Computer</i>
ROM	<i>Read-Only Memory</i>
SGBD	Sistema de Gerenciamento de Banco de Dados
SD	<i>Secure Digital</i>
SDHC	<i>Secure Digital High Capacity</i>
SGML	<i>Standard Generalized Markup Language</i>

SO:	Sistema Operacional
TCP:	<i>Transmission Control Protocol</i>
TKIP:	<i>Temporal Key Integrity Protocol</i>
UC:	Unidade de Controle
UDP:	<i>User Datagram Protocol</i>
ULA:	Unidade Lógica e Aritmética
UML:	<i>Unified Modeling Language</i>
URL:	<i>Uniform Resource Locator</i>
USB:	<i>Universal Serial Bus</i>

## SUMÁRIO

1	INTRODUÇÃO.....	7
1.1	Objetivo Geral .....	8
1.2	Objetivos Específicos .....	9
1.3	Justificativa.....	9
1.4	Organização do Trabalho.....	10
2	REFERENCIAL TEÓRICO .....	11
2.1	Plataformas de Hardware.....	11
2.1.1	Raspberry Pi 3 .....	12
2.1.2	Arduino Industrial 101 .....	14
2.1.3	Beaglebone Black.....	16
2.1.4	Node MCU ESP-12E .....	17
2.1.5	Comparação entre as plataformas de Hardware .....	19
2.2	Plataformas de Software .....	20
2.2.1	Java .....	20
2.2.2	Apache Tomcat.....	22
2.2.3	Biblioteca PI4J .....	23
2.2.4	PostgreSQL .....	23
2.2.5	JavaScript.....	24
2.3	Protocolos de Comunicação e Interoperabilidade.....	24
2.3.1	Protocolo de Comunicação 802.11 – WI-FI .....	25
2.3.2	HTTPS.....	30

2.3.3	JSON .....	32
2.3.4	M2M.....	32
2.3.5	MQTT .....	33
2.4	Trabalhos Relacionados.....	35
2.4.1	Arquitetura de IoT para automação residencial .....	36
2.4.2	Sistemas de automação residencial de baixo custo para redes sem fio.....	37
2.4.3	Comparativos com os trabalhos analisados .....	38
3	METODOLOGIA .....	40
3.1	Procedimentos de pesquisa .....	40
3.2	Referencial Teórico .....	41
3.3	Desenvolvimento dos protótipos .....	41
3.4	Processo de validação da proposta .....	42
4	DESENVOLVIMENTO .....	43
4.1	Aplicação Centralizadora .....	47
4.2	Aplicação Gerenciada .....	54
4.3	Protocolo .....	57
4.4	Segurança da Informação .....	61
4.5	Processo de Validação.....	64
5	CONCLUSÃO .....	66
5.1	Contribuições .....	67
5.2	Trabalhos futuros .....	68
	REFERÊNCIAS.....	69



## 1 INTRODUÇÃO

A *Internet of Thing*, IoT, compreende uma infraestrutura de rede com alta capacidade, baseada em protocolos de comunicação de rede que possuem atributos físicos e virtuais com suas próprias identidades (MIORANDI, 2012).

Segundo a previsão da US National Intelligence Council (NIC, 2008), a IoT tornou-se uma das tecnologias do futuro, estima-se que em 2025 estará presente em 25 bilhões de equipamentos, permitindo inúmeras oportunidades, tanto economicamente como tecnologicamente. Ainda conforme a NIC, devemos considerar duas incertezas, a agilidade do desenvolvimento e o ingresso no mercado (NIC, 2008). Segundo a International Data Corporation (IDC, 2017), projeta um crescimento de 15% anualmente, no período de 2015 até 2020, podendo chegar em aproximadamente 1,3 trilhões de dólares já em 2020.

A medida que a tecnologia avança, os dispositivos eletrônicos tornam-se mais baratos e em escalas menores, tendo assim capacidades e espaços limitados, tornado a IoT um novo paradigma de desenvolvimento de software, diferente dos paradigmas atuais (MIORANDI, 2012).

Com o desenvolvimento constante da IoT, tem-se dispositivos distribuídos geograficamente, ou seja, dispositivos e software, interligados a rede local de computadores, intranet, ou via rede mundial de computadores, a internet, comunicando-se e coordenando atividades entre si (COULOURIS, 2012).

Como pode ser constatado, a evolução da IoT é maximizada pela automação industrial e automação residencial. Segundo Muratoni (2013), a automação residencial é um conjunto de serviços proporcionados por sistemas tecnológicos integrados que

tem por finalidade satisfazer as necessidades mínimas em segurança, comunicação e conforto habitacional.

Sistemas interconectados naturalmente precisam seguir boas práticas de segurança da informação. De fato, sistemas distribuídos tendem a oferecerem pouca proteção da informação. Em alguns casos, o tráfego de informações pode ocorrer sem criptografia através da internet, expondo assim dados e informações sigilosas (TANENBAUM e STEEN, 2007).

Conforme Tanenbaum (2007), todos os recursos de sistemas distribuídos devem ser facilmente acessíveis, ocultando o fato de que os recursos estão espalhados pela rede. Todos esses recursos, devem serem implementados suportando as especificações previamente exigidas para os sistemas computacionais distribuídos.

Este trabalho, descreve o levantamento de requisitos de configurações de dispositivos da IoT, seguido pelo desenvolvimento de um protótipo, que faz a coleta de informações e envio de configurações. Esta implementação foi validada através da automação de um datacenter distribuído em duas filiais de uma empresa, monitorando a temperatura e gerenciando a execução de ações determinadas funções.

## **1.1 Objetivo Geral**

Este trabalho tem como objetivo geral desenvolver um protótipo dividido em duas aplicações, uma centralizadora e outra gerenciada, para permitir controlar dispositivos distribuídos. A aplicação centralizadora possibilita realizar a captação de informações de forma organizada, obedecendo protocolos de segurança da informação, protocolos de transporte, além de enviar configurações para a aplicação gerenciada, para permitir o desenvolvimento de recursos tecnológicos gerenciados a partir da aplicação da centralizadora.

## 1.2 Objetivos Específicos

- Implementação de uma aplicação centralizadora;
- Implementação de uma aplicação gerenciada pela aplicação centralizadora;
- Gerenciamento dos dados coletados dos dispositivos distribuídos;
- Utilização da biblioteca PI4J, para abstrair comunicação de baixo nível nos dispositivos gerenciados;
- Implementação do controle de segurança e encapsulamento via HTTP e HTTPS;
- Documentação e testes da aplicação centralizadora e da aplicação gerenciada.

## 1.3 Justificativa

O mercado de trabalho está sempre em constante atualização, cada vez mais, é procurado por profissionais com conhecimentos e capacidade que ultrapasse os conhecimentos obtidos na graduação. A área de IoT busca um profissional capaz de analisar e implementar soluções de comunicação entre plataformas computacionais e dispositivos, possibilitando configurar, captar e enviar informações para outros equipamentos interconectados (HASSELL, 2017).

O desenvolvimento de novas tecnologias apresenta um cenário, no qual os profissionais devem estar constantemente buscando por novos conhecimentos. Este trabalho possibilitará obter mais experiência, conhecimento, permitindo a atualização conforme as novas tecnologias e tendências de mercado. A IoT é uma tecnologia em expansão, por isso, é necessário utilizar ferramentas baseadas em software para maximizar a administração centralizada dos equipamentos distribuídos, facilitando a utilização e coleta de informações.

Este trabalho propõe uma solução que seja adaptativa e interconectada com outros dispositivos distribuídos da IoT, agrupando gerenciamento de dispositivos e centralização de informações.

## **1.4 Organização do Trabalho**

Para elucidar este trabalho, o mesmo é disposto em capítulos. O Capítulo 2 será apresentada uma visão geral sobre as Plataformas de Hardware, Software e Protocolos de Comunicação, apresentando alguns dispositivos e o software utilizado para o desenvolvimento do protótipo proposto.

No Capítulo 4 será apresentado o desenvolvimento do protótipo proposto neste trabalho, incluindo a implementação da aplicação centralizadora e aplicação gerenciada. No Capítulo 5 será apresentado o resultado deste trabalho.

## 2 REFERENCIAL TEÓRICO

Esse capítulo visa descrever as plataformas de hardware e software, mais utilizadas na literatura para dispositivos IoT, abordando também os protocolos de comunicação utilizados no desenvolvimento do trabalho.

As plataformas modulares de *hardware* podem oferecer recursos para torna-se um dispositivo IoT, para isso, são necessárias adaptações e implementações de recursos, sendo que alguns desses recursos são implementados via software.

Os softwares utilizados neste trabalho, foram escolhidos para trabalharem com a plataforma de *hardware* modular da Raspberry Pi 3.

A interoperabilidade, interconexão e a inteligência, são importantes para que os dispositivos da IoT possam funcionar. A plataforma de distribuição se da a protocolos de comunicação distribuídos.

Adicionalmente a seção 2.4, serão vistos alguns trabalhos relacionados que possuem similaridades com o trabalho proposto.

### 2.1 Plataformas de Hardware

Uma plataforma de hardware convencional, é constituída por uma *Central Processing Unit* (CPU), Memória *Dynamic Random-Access Memory* (DRAM) e Barramento (STALLINGS, 2002).

Por sua vez, a CPU, é a parte do dispositivo que busca e executa instruções computacionais. Conforme Stallings (2002), a CPU consiste de uma Unidade Lógica e Aritmética (ULA) e Unidade de Controle (UC). A ULA é designada a efetuar operações aritméticas, lógicas e demais operações relacionadas a CPU, por sua vez,

a UC, controla toda a execução das operações na CPU, trocando informações e sinais por meio de interfaces, como o barramento.

A Memória DRAM tem função de receber dados e instruções carregadas e endereçadas através dos registradores, para posteriormente serem processados ou executados na CPU.

O Barramento é um caminho na qual um ou mais dispositivos se comunicam, em um meio de transmissão compartilhado. Caso haja dois dispositivos que transmitem sinais ao barramento ao mesmo tempo, esses sinais irão se sobrepor. Para que a transmissão ocorra com sucesso, apenas um dispositivo pode transmitir sinais pelo barramento a cada instante (STALLINGS, 2002).

Toda a evolução da tecnologia, componentes e arquiteturas, permitiram que as plataformas de hardware se adaptem ao mercado da IoT, houve uma grande evolução no desempenho, consumo energético e dimensões dos equipamentos (MIORANDI, 2012).

Nas próximas seções serão apresentadas e avaliadas as principais plataformas de hardwares, detalhando as características de CPU, memória e barramento externo.

### **2.1.1 Raspberry Pi 3**

A Raspberry Pi é uma plataforma de hardware, criada pela Raspberry Pi Foundation. A Raspberry Pi é um equipamento que contém características de um sistema computacional de alta eficiência (RICHARDSON, WALLACE, 2013), o qual, conforme especificações do fabricante, consome 12 W de energia.

Uma das características importante da Raspberry PI, são as suas 40 portas *General Purpose Input/Output* (GPIO), que possuem as funcionalidades de entrada e saída e que não possuem atribuições dedicadas, permitindo assim serem manipuláveis.

O Raspberry Pi é um dos poucos dispositivos embarcados que não foi introduzido um relógio de tempo real, permitindo obter a data e hora somente através

do protocolo de sincronização de relógios, assim o funcionamento da Raspberry Pi, só ocorre quando o equipamento estiver operando.

Na Figura 1 é possível visualizar a Raspberry Pi 3, com dimensões de 8,5 cm de profundidade, 5,6 cm de largura e 1,7 cm de altura.

Figura 1 – Raspberry Pi 3



Fonte: Raspberry Pi Foundation 2017

As principais especificações da Raspberry Pi 3 são listadas a seguir:

- 4 núcleos ARM Cortex A53 operando a 1.2 Ghz;
- 1 GB de memória RAM LPDDR2 (*Low Power Double Data Rate 2*) de 900 MHz;
- Slot para cartão SDHC (*Secure Digital High Capacity*) de até 64 Gb;
- 4 USBs 2.0;
- 40 Pinos GPIO
- HDMI (*High Definition Multimedia Interface*);
- CSI (*Serial Interface*) e DSI (*Display Serial Interface*);
- Wireless de 2.4 Ghz;
- Ethernet 10/100 Mbit/s.

O Sistema Operacional é instalado diretamente no cartão SD<sup>1</sup>, introduzido na Raspberry Pi, por este motivo, o cartão SD recomendado pela fabricante é o Kingston SDA10/16 GB, o qual suporta 90 MB por segundo de escrita e 45 MB por segundo de gravação (KINGSTON, 2017). O Sistema Operacional recomendado é o Raspbian, uma distribuição Linux originada da Distribuição Linux Debian.

O Raspbian é uma versão minimalista do Linux criado pela comunidade de Software Livre<sup>2</sup>, mantida pela organização Raspbian. Este sistema operacional foi otimizado para o hardware da Raspberry Pi.

### 2.1.2 Arduino Industrial 101

Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software. O Arduino é capaz de ler entradas e transformá-lo em uma saída<sup>3</sup>. Todas as suas plataformas de hardware possuem um microcontrolador capaz de interpretar e executar instruções de código. Para desenvolver funções, é necessário utilizar a linguagem de programação Arduino.

Na Figura 2 é possível visualizar a plataforma Arduino Industrial 101, com dimensões de 4,2 cm de profundidade, 5,1 cm de largura e 1,5 cm de altura.

Figura 2 – Arduino Industrial 101

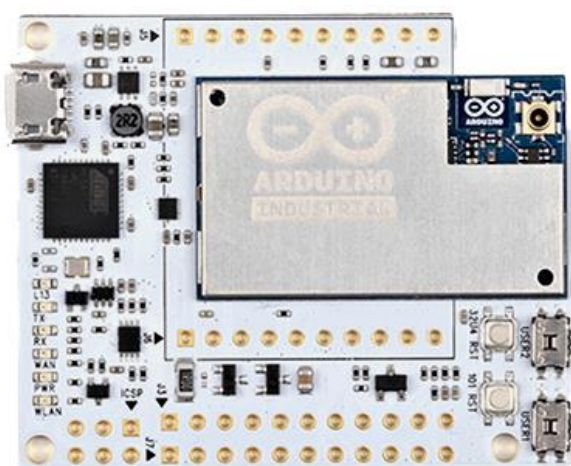
---

<sup>1</sup> SD é o termo utilizado para designar os Cartões de Memória não voláteis, *Secure Digital*.

<sup>2</sup> Software Livre é a comunidade que tem como objetivos por concessão de direitos de liberdade, podendo aos usuários: executar, copiar, distribuir, estudar, modificar o software.

<sup>3</sup> Arduino Foundation. Introduction of the Arduino. 2017. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction>> Acesso em: 27 fev. 2017





Fonte: Arduino Foundation 2017

As principais especificações do Arduino Industrial 101 são listadas a seguir:

- Atheros AR9331 operando a 400 Mhz;
- Microcontrolador ATmega32u4 de 16 Mhz;
- 64 MB de memória RAM LPDDR2 (*Low Power Double Data Rate 2*);
- 1 USBs 2.0;
- 20 Pinos I/O Digital;
- 12 Pinos Analógicos;
- Wireless de 2.4 Ghz;
- Ethernet 10/100 Mbit/s.

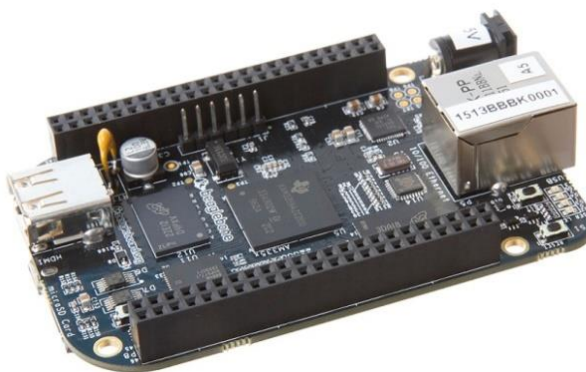
A Arduino Industrial 101, é uma placa específica para IoT, conforme a fabricante, o dispositivo é considerado de baixo custo, baixo consumo e com recursos de *wireless* integrados, facilitando a conexão de rede e Internet (ARDUINO, 2017).

Conforme a Arduino Foundation, a Arduino Industrial 101, possui uma tecnologia chamada, Arduino Ciao. Esta tecnologia permite estes dispositivos comunicarem-se diretamente com alguns serviços, protocolos de comunicação e conectar-se a redes sociais. Ainda conforme a Arduino Foundation, o Ciao foi projetado e desenvolvido para ser facilmente configurável, com os recursos do sistema. Seu objetivo é facilitar o desenvolvimento de funções com poucas linhas de comando (ARDUINO, 2017).

### 2.1.3 Beaglebone Black

A Beaglebone black é uma plataforma de hardware criada pela Beagle Board Foundation. A Beaglebone black, semelhante a Raspberry Pi 3, possui mecanismos de alta eficiência, unindo processador de alto desempenho com baixo consumo, além de um baixo custo (Beagle Board, 2017). Na Figura 3 é possível visualizar a BeagleBone Black, com dimensões de 8,65 cm de profundidade, 5,35 cm de largura e 1,9 cm de altura.

Figura 3 – Beaglebone Black



Fonte: Beagle Board Foundation 2017

As principais especificações da Beaglebone Black são listadas a seguir:

- Sitara AM3359AZCZ100 Cortex A8 operando a 1 Ghz;
- 512 MB de memória RAM LPDDR3 (*Low Power Double Data Rate 2*);
- 1 USBs 2.0;
- Slot para cartão SDHC (*Secure Digital High Capacity*) de até 32 Gb;
- 65 Pinos I/O Digital;
- 7 Pinos Analógicos;
- Ethernet 10/100 Mbit/s;
- HDMI (*High Definition Multimedia Interface*).

A Beaglebone Black, permite a instalação de diversos Sistemas Operacionais, Debian, Ubuntu, Android, Windows 10 e outros sistemas minimalistas, como por exemplo, o Angström e o Poky Linux.

O Angström e o Poky Linux, são distribuições Linux, especialmente desenvolvidas para dispositivos embarcados, ambos os projetos são mantidos pela Linux Foundation, com projeto denominado Projeto Yocto.

#### 2.1.4 Node MCU ESP-12E

O Node MCU, é uma plataforma de distribuição aberta, é baseado em um módulo ESP8266, cujo a sua fabricante é a empresa, Espressif, fabricante de semicondutores, de baixo consumo e custo, especialmente soluções para aplicações IoT<sup>4</sup>.

O Node MCU é um dispositivo muito compacto, largamente utilizado em IoT, especialmente em automação residencial. O dispositivo possui CPU para interpretação e execução das instruções de código, operando em uma frequência de 80 Mhz, podendo ser desenvolvida na Plataforma Arduino ou na Linguagem de programação LUA.

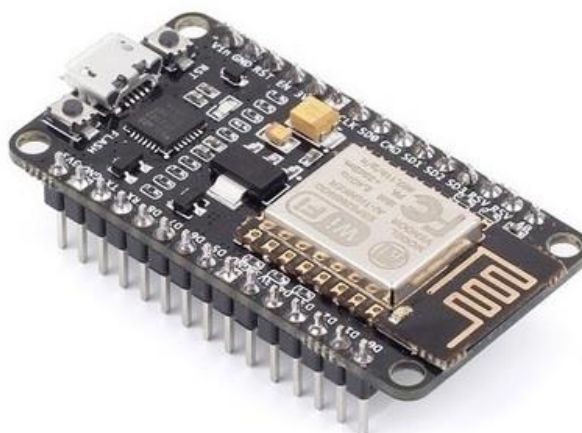
Conforme o fabricante, o consumo é extremamente baixo, podendo chegar até 0,6 W de energia por hora, operando de forma normal. O módulo permite a função de *deep sleep*, ou seja, sono profundo, assim o dispositivo só realiza operações quando for acionado, voltando ao estado de *deep sleep* após a execução das instruções programadas.

Na Figura 4 é possível visualizar a placa do Node MCU, com dimensões de 4,9 cm de profundidade, 2,5 cm de largura e 0,7 cm de altura.

---

<sup>4</sup> Espressif Company. Products. 2017. Disponível em:  
<<http://espressif.com/en/products/hardware/esp32-devkitc/overview>> Acesso em: 09 mai. 2017.

Figura 4 – Node MCU ESP-12E



Fonte: Espressif Company 2017

As principais especificações da Node MCU são listadas a seguir:

- CPU RISC operando a 80 Mhz;
- 32 KB de memória RAM para instruções;
- 96 KB de memória RAM para dados;
- 64 KB de memória ROM para *boot*;
- 1 Micro USB 2.0;
- 11 Portas GPIO;
- Conversor Analógico / Digital;
- Wireless de 2.4 Ghz.

Como pode-se observar pelas especificações, a conexão *wireless* deste dispositivo, pode conectar-se a outros equipamentos para enviar/receber informações, além disso, os pinos GPIO podem proporcionar controle de diversos sensores conectados (NODE MCU, 2017).

### 2.1.5 Comparação entre as plataformas de Hardware

Conforme apresentado na seção 2.1, existem várias opções de hardware ao trabalhar com IoT e Computação Distribuída, todas as plataformas possuem vantagens e desvantagens, na Tabela 1 serão apresentadas as especificações das plataformas já apresentadas.

Tabela 1 – Especificações das plataformas apresentadas

Equipamento	CPU	Memória	Conexão	Suporte a Java/JSP	GPIO e Portas Analógicas
Raspberry Pi 3	Cortex A53 1,2 Ghz	1 GB LPDDR2	Wireless de 2,4 Ghz e Ethernet Mbit/s	Sim	40 Pinos Digitais, não há pinos analógicos
Arduino Industrial 101	Atheros AR9331 400 Mhz	64 Mb LPDDR2	Wireless de 2.4 Ghz	Não	20 Pinos digitais, 12 Pinos analógicos.
Beaglebone Black	Cortex A8 1 Ghz	512 Mb LPDDR3	Ethernet 10/100 Mbit/s	Sim	65 Pinos digitais, 7 pinos analógicos.
Node MCU ESP-12E	CPU Risc 80 Mhz	96 Kb Memória RAM	Wireless de 2.4 Ghz	Não	11 Pinos digitais, conversor Analógico/Digital.

Fonte: Autor, 2017

Em modo geral, precisa-se avaliar também a necessidade de desempenho, consumo energético e utilização dos pinos de extensão fornecidas por cada plataforma. Neste presente trabalho, será utilizado a Raspberry Pi 3, de acordo com as especificações demonstradas, o Raspberry Pi 3, possui velocidade de processamento de 1,2 Ghz, 1 GB de memória RAM e quantidades de pino de extensão suficientes para realizar o desenvolvimento do protótipo.

## **2.2 Plataformas de Software**

Uma plataforma de software é constituída pela elaboração de algoritmos. Os algoritmos, descrevem passo a passo como uma instrução ou tarefa deve ser realizada. As instruções devem seguir uma lógica, na qual visa a conclusão da tarefa a proposta na elaboração do algoritmo. Um conjunto de algoritmos será codificado em uma linguagem de programação, na qual após a compilação, ou não, sucede-se o software (MENDES, 2009).

Para Silveira et al. (2011), um software não é feito apenas da elaboração de algoritmos, pelo contrário, necessita-se de uma estruturação e arquitetura para manter uma evolução constante das lógicas implementadas.

Conforme Fernandes, 2017, um software está subdivido em partes, são elas: Instruções legíveis para máquina, seus componentes, dados e conteúdo audiovisual.

As instruções legíveis para máquina implicam que a máquina possa entender e ler as instruções fornecidas para tomar uma ação para realizar determinada função a que lhe compete. Os componentes podem ser constituídos por outros programas menores.

O software pode conter dados, o que significa que ele não é composto apenas por instruções. A esse respeito Fernandes (2017), declara: “A existência de dados em um programa está compatível com a definição do Modelo Computacional de von Neumann, onde um programa armazenado na memória é formado por instruções e dados”.

Nas próximas seções, serão apresentadas algumas tecnologias utilizadas no desenvolvimento deste trabalho.

### **2.2.1 Java**

A tecnologia Java foi criada pela Sun Microsystems em 1991, criado a partir de um pequeno projeto e secreto chamado “The Green Project”. O projeto foi liberado por James Gosling, com uma equipe formada de 13 pessoas (SERSON, 2007).

A linguagem de programação Java foi totalmente ampliada, testada e experimentada por mais de quatro milhões de desenvolvedores de software. A tecnologia Java adquiriu um valor inestimável, permitindo aos desenvolvedores compilar um software e executá-lo em qualquer plataforma; criar programas para execução em navegadores; escrever aplicativos potentes e eficazes para celulares, dispositivos de redes (SERSON, 2007). Acompanhando todas essas características, o Java é composto por uma linguagem de programação e de uma plataforma (API e a máquina virtual).

Conforme Serson (2007), as principais características da linguagem de Programação Java listadas a seguir:

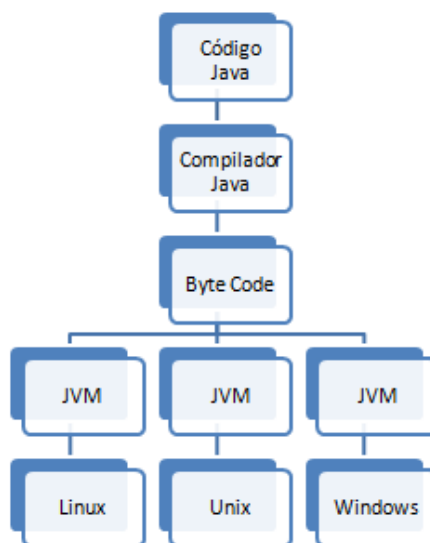
- Orientada a Objetos;
- *Multithread*;
- Singularidade e Interpretável;
- Portável;
- Robusta;
- Segura;
- Alto Desempenho.

A *Java Virtual Machine* (JVM), é o programa que recebe e executa os algoritmos da aplicação Java, convertendo em *bytecodes* códigos executáveis para a máquina. Sua função é gerenciar a execução dos aplicativos, à medida que são executados (MENDES, 2009).

Segundo Mendes (2009), a portabilidade a linguagem de programação Java, consolidou-se em diversas plataformas pela atribuição da JVM, tornando um arquivo já compilado a executar em qualquer dispositivo em que a JVM esteja disponível.

Na Figura 5 é demonstrado o processo de compilação do arquivo .java, contendo o código escrito, após a compilação, é gerado um arquivo que será executado pela JVM, permitindo a compatibilidade para execução em diversos Sistemas Operacionais.

Figura 5 – Etapa de compilação



Fonte: Adaptado de MACHADO, 2017

Neste trabalho será utilizado a linguagem de programação JAVA, visto que seu ambiente de execução é multiplataforma, é robusto e escalável, conforme enfatizado na seção 2.2.1.

### 2.2.2 Apache Tomcat

O Tomcat é um servidor Web, que implementa as tecnologias Java Servlet e JavaServer Pages (JSP). Esta tecnologia é mantida e disponibilizada pela Apache Software Foundation na licença de Software Livre. O Tomcat, é um servidor Web escalável, permite a conexão de múltiplos *browsers* através do protocolo *Hypertext Transfer Protocol* (HTTP) (APACHE, 2017). O Apache Tomcat é baseado na linguagem Java, que é multi-plataforma.



### 2.2.3 Biblioteca PI4J

O Projeto PI4J, tem como propósito de fornecer uma *Application Programming Interface* (API), orientada a objetos, para acessar de forma fácil e rápida as funcionalidades de *Input/output* (I/O), além de abstrair a integração e interrupções de software de baixo nível, disponibilizando facilidade para implementação das lógicas de negócios (PI4J, 2017).

Na Código 1, é apresentado um exemplo de código fonte, com a finalidade de ativar a saída 01 durante 5 segundos, após o sinal de saída para a porta 01 será interrompido.

Código 1 – Código exemplo ativando porta de saída 01.

```
public static void main(String[] args) throws InterruptedException {
    final GpioController gpio = GpioFactory.getInstance();
    final GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(RaspiPin.GPIO_01,
                                                                    "Teste", PinState.HIGH);

    Thread.sleep(5000);
    pin.low();
    gpio.shutdown();
}
```

Fonte: Autor, 2017

É possível observar a abstração que a biblioteca fornece, assim sendo possível enviar um sinal digital para uma determinada saída do dispositivo com poucas instruções de código, possibilitando código fonte legível.

### 2.2.4 PostgreSQL

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) relacional. Conforme Milani (2008), o PostgreSQL é uma ferramenta que é responsável pelo armazenamento de dados e gerenciamento das informações de acordo com as regras pré-estabelecidas.

O SGBD, implementa diversos recursos para suportar bases que necessitam de alta disponibilidade. Alguns dos recursos suportados são: replicação síncrona e assíncrona, suporte para *Procedural Language Structured Query Language* (PL/SQL) (POSTGRESQL, 2017).

### 2.2.5 JavaScript

JavaScript é uma linguagem de programação interpretada, foi criada por Brendan Eich, e inspirada pela linguagem script ECMAScript, padronizada pela norma ECMA-262 e mantida pela ECMA International (ECMA, 2016). Projetado para possibilitar que os navegadores de internet pudessem executar *scripts* no computador do cliente, com interação do usuário sem que o *script* seja executado pelo servidor, fazendo assim, comunicação assíncrona, alterando o conteúdo exibido na página Web, conforme especificado nas normas ECMA-262 (ECMA, 2016).

Atualmente é uma das principais linguagens para programação *front end* para páginas da Web, utilizando JQuery, biblioteca de funções que interage com o *HyperText Markup Language* (HTML), por fim, o Bootstrap, é um *framework* que permite interagir com modelos de *design* e estilos, permite usar componentes reutilizáveis, tudo isso para prover a melhor implementação de *layout* para páginas da Web.

## 2.3 Protocolos de Comunicação e Interoperabilidade

Um protocolo de comunicação é uma forma de comunicação que é composto por diversas regras para coordenar a troca de dados entre indivíduos de um sistema. Os protocolos de comunicação mais comuns são a internet, *WiFi*, GSM, 3G, 4G, *bluetooth*, indispensáveis nos dias de hoje (COULOURIS, 2012).

A interoperabilidade é a capacidade de realizar a comunicação por meio de diferentes sistemas ou protocolos, reutilizando informação compartilhada. Conforme Coulouris (2012), para um software seja considerado Inter operável, é necessário

possuir alguns atributos. Os atributos devem possuir a capacidade para troca de informações com diferentes sistemas da informação, uniformidade e padronização, por fim, simplicidade da comunicação.

Coulouris (2012), ressalta a importância da interoperabilidade em sistemas distribuídos quando constatou que o crescimento acelerado de aplicações para redes e sistemas distribuídos é resultado do uso de computadores e que para a utilização destes, dependem do compartilhamento de dados entre programas executados em diferentes estações de trabalho.

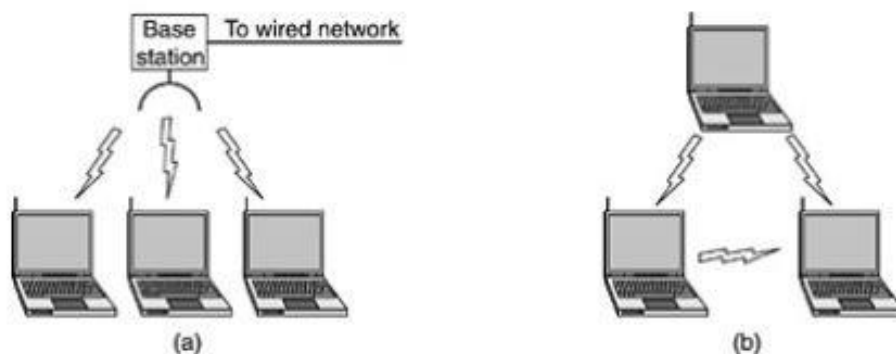
Na seção 2.3.1, será apresentado o protocolo 802.11, o *Wi-Fi*, principal protocolo de transporte utilizado neste trabalho.

### **2.3.1 Protocolo de Comunicação 802.11 – *WI-FI***

O protocolo Wi-Fi foi estabelecido pela WI-FI Alliance e desempenha os padrões da norma 802.11. A norma compreende as especificações e padrões da camada física e de enlace, com a finalidade da implementação de uma rede local sem fio, nas frequências de 2.4 GHz e 5 GHz.

No padrão Wi-Fi, são permitidos dois tipos de arquitetura: Estação Base e Ad hoc. No primeiro caso, toda a comunicação é realizada através da estação base. No segundo caso, os computadores podem estabelecer comunicação sem envolver pontos de acesso centrais ou estações bases. A Figura 6 demonstra os dois tipos de arquitetura apresentado anteriormente.

Figura 6 – A Figura (a) apresenta uma Rede sem fio com uma estação base. A Figura (b) apresenta uma Rede tipo Ad hoc.



Fonte: TANENBAUM, 2011

### 2.3.1.1 Camada Física

Conforme o modelo OSI, a camada física compreende o meio físico, a transmissão de dados no canal de comunicação. O meio físico pode ser dividido em três meios: guiado, sem fio e satélite (TANENBAUM, 2011). A camada física no *Wi-Fi*, meio sem fio, necessita de alguns parâmetros, a modulação do sinal e a frequência da banda utilizada. Desta forma, o *Wi-Fi*, pode ser dividido em duas faixas de operação: 2.4 GHz e 5 GHz.

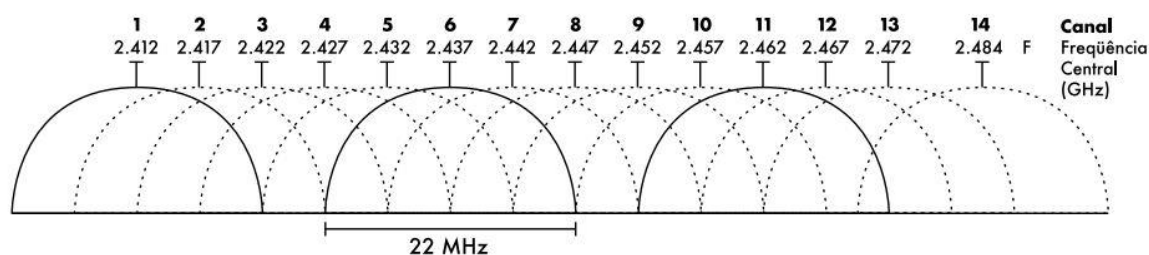
A primeira faixa opera entre 2.4 GHz a 2.5 GHz, abrangendo assim 14 canais de operações, com largura de 22 Mhz, mas estão separados por apenas 5 MHz. Isso significa que há intersecção entre canais adjacentes, eles podem interferir um com o outro (BUTLER, 2013). O primeiro canal está centralizado na frequência de 2,412 GHz e o último canal em 2,484 GHz. Na Figura 7, é exibida a intersecção dos canais adjacentes, é possível verificar que não há interferências entre os canais 1, 6 e 11.

A segunda faixa encontra-se entre 5,150 GHz a 5,850 GHz. Elas são subdivididas em três modos de operação: indoor, indoor/outdoor e outdoor. As frequências indoors, trabalham entre 5,150 GHz e 5,250 GHz, as frequências

indoors/outdoors, trabalham entre 5,250 GHz e 5,350 GHz, por fim, as frequências outdoors, trabalham entre 5,725 GHz e 5,825 GHz.

Apesar das definições apresentadas, cada região possuem um órgão regulamentador para utilização dessas frequências, dessa forma, as regras e distribuição de frequências podem variar entre as regiões.

Figura 7 – Canais e Frequências 802.11b



Fonte: BUTLER, 2013

### 2.3.1.2 Camada Enlace

A camada de enlace de dados executa diversas funções específicas. Dentre elas, fornecimento de uma interface de serviço para a camada de rede, suportar erros de transmissão, controlar o fluxo de dados, de tal forma que receptores lentos não sejam apressados por transmissores rápidos (TANENBAUM, 2011).

Conforme Tanenbaum, para lidar com os problemas no protocolo 802.11, é adotado dois modos de operação. O primeiro é chamado de *Distributed Coordination Function* (DCF), que não usa nenhuma espécie de controle central, tornando assim, semelhante ao padrão *Ethernet*. O segundo é chamado de *Point Coordination Function* (PCF), utiliza a estação base para controlar, tornando-se opcional a sua implementação.

### 2.3.1.3 Revisões da norma IEEE 802.11

Com o passar do tempo e a evolução dos sistemas computacionais e sistemas *mobile*, houve a necessidade de implementar melhorias de desempenho, qualidade

da transmissão e aumento da potência do sinal. As alterações das especificações ocorreram na largura de banda, frequência utilizada, modulação do sinal, abrangendo consequentemente na taxa de transmissão.

#### **2.3.1.3.1 IEEE 802.11a**

Foi concebido após os padrões 802.11 e 802.11b. Permite alcançar velocidades de até 54 Mbps dentro dos padrões estabelecidos pela IEEE. Esse padrão opera na frequência de 5 GHz. As suas principais vantagens são a velocidade e a ausência de interferências. Uma das suas desvantagens é o menor alcance (IEEE, 2012).

#### **2.3.1.3.2 IEEE 802.11b**

Lançado em 1999, este padrão alcança uma taxa de transmissão de até 11 Mbps. Esse padrão opera na frequência de 2.4 GHz (IEEE, 2012). Rapidamente popularizou-se pelo baixo preço dos seus dispositivos. Neste padrão há alta interferência de outra transmissão, pois a 2.4 GHz, que é equivalente aos telefones móveis, fornos micro-ondas e dispositivos Bluetooth.

#### **2.3.1.3.3 IEEE 802.11g**

Esta revisão permitiu a retro compatibilidade com os dispositivos que implementavam a norma 802.11b. A velocidade oferecida nesse padrão é de até 54 Mbps. Esse padrão opera dentro da frequência de 2,4 GHz. De acordo com Butler (2013), essa revisão foi muito importante, evoluiu a criptografia da rede, com métodos de autenticação WEP estática e WPA (Wireless Protect Access) com criptografia, com os métodos *Temporal Key Integrity Protocol* (TKIP) e *Advanced Encryption System* (AES) (IEEE, 2012).

#### 2.3.1.3.4 IEEE 802.11n

A principal melhoria no padrão 802.11n é a utilização de dois canais simultâneos para a comunicação, trazendo um melhor desempenho e confiabilidade. Esse padrão opera dentro das frequências de 2.4 GHz e/ou 5 GHz. A taxa de transferência oferecidas nesse padrão são de 65 Mbps a 450 Mbps (IEEE, 2012).

#### 2.3.1.3.5 IEEE 802.11ac

Esse padrão opera em faixa de 5 GHz, esse padrão oferece uma largura de canal maior de até 160 MHz. Projetado para conexão em altas taxas de transferências, superando a faixa de 1 Gbps, padronizando inicialmente em 1.3 Gbps (IEEE, 2013).

Como verificado na seção 2.3.1.3, abaixo uma breve tabela comparativa entre as revisões da norma IEEE 802.11, na Tabela 2 será apresentado uma breve comparação entre as revisões, demonstrando a taxa de transferência alcançada em cada revisão, juntamente com a frequência da transmissão.

Tabela 2 – Padrões IEEE com seus principais aspectos

Padrão	Mbps (máximo)	Frequência (GHz)
802.11a	54 Mbps	5 GHz
802.11b	11 Mbps	2.4 GHz
802.11g	54 Mbps	2.4 GHz
802.11n	450 Mbps	2.4 GHz e 5 GHz
802.11ac	1.3 Gbps	5 GHz

Fonte: Autor, 2017

### 2.3.2 HTTPS

O protocolo *Hypertext Transfer Protocol Secure* (HTTPS), foi concebido para gerenciar e trabalhar com a transferência de páginas da Web de forma segura. O protocolo é implementado em aplicações como, navegadores da Web e Servidores da Web. Contudo, o HTTPS também é utilizado em sistemas que não estão ligados à Web (TANEMBAUM, 2007).

O protocolo funciona através de troca de mensagens entre cliente e servidor, essa troca de mensagens é criptografada através de certificados. A mensagem de requisição ou de retorno é composta por uma linha inicial, nenhuma ou mais linhas de cabeçalhos, uma linha em branco obrigatória, assim finalizando o cabeçalho e por último o corpo da mensagem (W3C, 2017).

As mensagens são enviadas e respondidas através de métodos de solicitação, as instruções mais utilizadas são: GET e POST.

O método GET solicita um retorno do recurso especificado, tem como objetivo apenas recuperar dados, normalmente é incluída na *Uniform Resource Locator* (URL), os parâmetros para serem lidos no servidor. O Código 3 apresenta uma requisição HTTP enviada pelo cliente, logo abaixo, no Código 3 é apresentado um breve trecho da resposta do servidor.

#### Código 2 – Código exemplo de uma requisição GET

```
public static void main(String[] args) throws MalformedURLException, IOException {
    String urlString = "http://univates.br/";
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("POST");
    BufferedReader rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    String line;
    while ((line = rd.readLine()) != null) {
        System.out.println(line);
    }
    rd.close();
}
```

Fonte: Autor, 2017



### Código 3 – Retorno do servidor

```
<!DOCTYPE html>
<html>
<head prefix="og: http://ogp.me/ns#">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta name="viewport" content="width=device-width, initial-scale=1"/>
<title>Univates</title></html>
```

Fonte: Autor, 2017

O método POST submete dados para serem processados pelo recurso encontrado no servidor. Os dados que são enviados através do protocolo, são incluídos no corpo do HTTPS, diferente do método GET que inclui informações diretamente na URL do navegador.

Como demonstrado na seção 2.3.2, a Tabela 3 – Métodos de solicitação HTTPS, demonstra uma breve comparação a algumas características que ambos os métodos utilizam. Para efeito deste trabalho, foram selecionados os quesitos de cache, histórico, tamanho dos dados e segurança.

Tabela 3 – Métodos de solicitação HTTPS

Solicitação / Informações	GET	POST
Cache	Pode ser armazenado em cache	Não armazenado em cache
Histórico	Os parâmetros permanecem no histórico do navegador	Os parâmetros não são salvos no histórico do navegador
Tamanho dos dados	Até 2048 caracteres	Sem limite
Segurança	Menos seguro, os dados fazem parte da URL	Não são armazenados no histórico ou em logs no servidor Web

Fonte: Autor, 2017

### 2.3.3 JSON

O *JavaScript Object Notation* (JSON), foi concebido com o objetivo promover legibilidade dos atributos e valores a humanos, também está estruturado para fácil interpretação pelas máquinas. JSON é utilizado em transmissões de objetos contendo pares de atributos e valores.

Ainda conforme Silva (2009), a finalidade do projeto JSON, facilitaria a implementação e interoperabilidade entre sistemas computacionais, os mesmos podem fazer troca de dados de forma legível. No Código 4 é apresentado um exemplo de código fonte.

#### Código 4 – Estrutura dos atributos e valores JSON

```
{ "Dispositivos" :
  [
    { "Raspberry pi": "192.168.2.2", "Memória": [1024] },
    { "Beaglebone Black": "192.168.2.3", "Memória": [512] },
    { "Arduino Industrial": "192.168.2.4", "Memória": [ 64 ] }
  ]
}
```

Fonte: Autor, 2017

### 2.3.4 M2M

M2M é a sigla para *Machine-to-Machine*. M2M possui um papel de estabelecer condições para que dispositivos possam trocar informações, bilateralmente, através de uma rede de comunicação (BOSWARTHICK, 2012).

O M2M engloba dispositivos que possuem a capacidade de coletar informações em um determinado local. Estes dispositivos estabelecem uma comunicação, por meio de uma conexão cabeada ou não, para assim transmitir os dados coletados. Estes dispositivos, podem serem incapazes de realizar a comunicação, executando assim a comunicação com outro dispositivo que agirá com um intermediário, para assim transmitir os dados recebidos até o destino (BOSWARTHICK, 2012).

Desta maneira, pode-se definir M2M, como um composto de tecnologias que possuem o objetivo de estabelecer a comunicação entre os dispositivos com capacidade limitada ou nula, sem que um ser humano interaja da comunicação e transmissão.

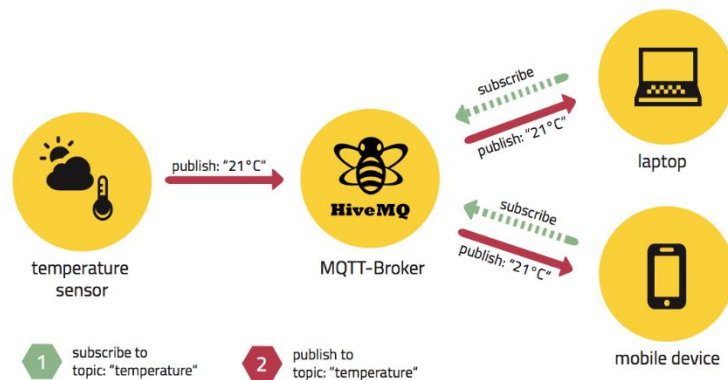
### 2.3.5 MQTT

O *Messaging Queue Telemetry Transport* (MQTT), criado por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech) em 1999. O MQTT, é um protocolo de troca de mensagens, baseado em uma arquitetura denominada *publish and subscribe*, direcionado para redes de baixa segurança e dispositivos com capacidade limitada (BARROS, 2015).

Conforme Hivemq (2015), o central da comunicação do protocolo MQTT é o *broker*. Todas as mensagens trocadas entre o remetente e o destinatário são transferidas através dele. Cada cliente que envia uma mensagem para o *broker* inclui uma informação na mensagem. A informação chega até o *broker*. Quando o cliente deseja receber uma informação, envia para o *broker* uma requisição. Portanto, esse protocolo permite uma solução altamente escalável, sem dependências entre os sensores de dados e os clientes, que desejam receber a informação.

A Figura 8, representa o funcionamento do protocolo MQTT, o envio de informações do sensor até o *broker* e do *broker* até o cliente final, enviando e recebendo as mensagens através das subscrições e publicações até o cliente.

Figura 8 – Protocolo MQTT



Fonte: HIVEMQ, 2017

O Código 6, demonstra um código de exemplo para conexão entre um sensor e o *broker*. Note que a URL de destino para o broker, é uma conexão *Transmission Control Protocol* (TCP) direcionado a uma porta de conexão, a porta 1883.

Código 5 – Realizando uma conexão com o *broker*

```
public class Publisher
{
    public static final String BROKER_URL = "tcp://broker.mqttdashboard.com:1883";
    private MqttClient client;

    public Publisher()
    {
        String clientId = Utils.getMacAddress() + "-pub";
        try
        {
            {
                client = new MqttClient(BROKER_URL, clientId);
            }
            catch (MqttException e)
            {
                {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Fonte: HIVEMQ, 2017

O Código 6, apresenta um código, cujo sua função é publicar a temperatura obtida em um determinado sensor, nesse exemplo é representado por um número randômico gerado pelo algoritmo.

Código 6 – Realizando uma publicação de temperatura para o *broker*

```
public static final String TOPIC_TEMPERATURE = "home/temperature";
while (true)
{
    publishTemperature();
    Thread.sleep(500);
}

private void publishTemperature() throws MqttException {
    final MqttTopic temperatureTopic = client.getTopic(TOPIC_TEMPERATURE);
    final int temperatureNumber = Utils.createRandomNumberBetween(20, 30);
    final String temperature = temperatureNumber + "°C";
    temperatureTopic.publish(new MqttMessage(temperature.getBytes()));
}
```

Fonte: HIVEMQ, 2017

## 2.4 Trabalhos Relacionados

Com a finalidade de avaliar a viabilidade dos protótipos, foram realizadas buscas por trabalhos concluídos semelhantes a este. Nesta seção, serão demonstrados dois trabalhos relacionados a arquitetura de IoT e automação residencial. A análise dos trabalhos de conclusão citados na seção 2.4.1 e 2.4.2, favoreceu as definições dos Objetivos Específicos encontrados na seção 1.2.

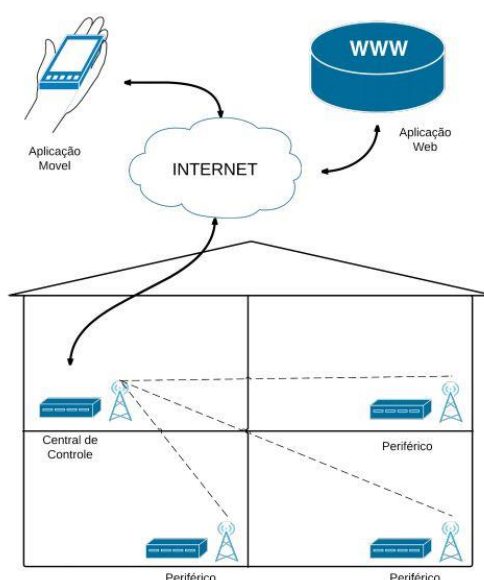
### 2.4.1 Arquitetura de IoT para automação residencial

O primeiro trabalho analisado, é a proposta desenvolvida por Barboza (2015), visa a implementação e utilização de uma arquitetura de internet das coisas para automação residencial.

A arquitetura desenvolvida por Barboza (2015), permite a adição de dispositivos residenciais no protótipo, nesses dispositivos encontram-se os periféricos, acessar o cadastro de dispositivos, manipulação das informações referentes aos dispositivos cadastrados, por fim, monitoramento e controle dos sensores cadastrados nos periféricos. Todo o cadastro de dispositivos é feito numa interface Web, o aplicativo *mobile* tem por objetivo realizar consultas rápidas via Internet.

Conforme Barboza (2015), a aplicação Web e a aplicação *mobile*, realizam a conexão diretamente com o dispositivo, que por ele é chamado de Central de controle, permitindo assim alterar configurações e receber informações dos periféricos. Na Figura 9 é apresentado a arquitetura do sistema, conforme planejado por Barboza (2015).

Figura 9 – Arquitetura do sistema, Barboza (2015)



Fonte: BARBOZA, 2015

### 2.4.2 Sistemas de automação residencial de baixo custo para redes sem fio

O segundo trabalho analisado, é a proposta desenvolvida por Silva (2014), realizou o desenvolvimento de uma aplicação cliente/servidor para realizar automação residencial. Essa automação, permite a leitura e alteração dos sensores pré-configurados na aplicação.

A implementação da comunicação entre os dispositivos e sensores, foram realizados através de uma rede *Wi-Fi*, obedecendo o protocolo de comunicação desenvolvido pela Zigbee Alliance. Sua proposta tem a finalidade de utilizar uma comunicação de baixo consumo energético para interligar sensores de movimento, gás, temperatura e luminosidade, conectados a uma placa Arduino Mega, controlados por um aplicativo Android.

De acordo com Silva (2014), a arquitetura de sistema apresentada, é uma área de comunicação através do protocolo Zigbee, os sensores são conectados a uma estação base, o Xbee *coordinator*, que é responsável por trocar informações entre os sensores e o Arduino. Por fim, o celular conecta-se a uma estação base, trocando informações com o Arduino. Na Figura 10 é apresentado a arquitetura do sistema desenvolvido por Silva (2014).

Figura 10 – Arquitetura do sistema, Silva (2014)



Fonte: SILVA, 2014

### 2.4.3 Comparativos com os trabalhos analisados

Na Tabela 4, são apresentadas as informações e tecnologias abordadas nos trabalhos analisados, além da demonstração das informações presentes neste trabalho. As principais informações a serem comparadas entre os trabalhos são: os tipos de dispositivos, tipo de rede para interconexão, protocolos de comunicação e por fim, segurança da informação.

Tabela 4 – Comparativo entre os trabalhos analisados

Autor / Tecnologia	Barboza (2014)	Silva (2014)	Santin (2017)
Dispositivos	Arduino	Arduino	Raspberry PI 3
Rede	Wi-Fi	Zigbee e Wi-Fi	Ethernet e Wi-Fi
Tipo de Rede	IEEE 802.11 a/b/g/n	IEEE 802.11 a/b/g/n e IEEE 802.15.4	IEEE 802.11 a/b/g/n e IEEE 802.3
Tipo de transmissão	HTTP	HTTP e JSON	HTTP ou HTTPS e JSON
Segurança da informação	Não	Não	HTTPS
Propriedades e Atributos	Fixas na aplicação de controle	Fixas na aplicação <i>mobile</i>	Propriedades e atributos dinâmicos definidos pelo dispositivo gerenciado.

Fonte: Autor, 2017

De acordo com as informações apresentadas na Tabela 4, as tecnologias utilizadas nos trabalhos apresentados, são semelhantes. Conforme apresentado na seção 2.

Uma das principais características que será desenvolvida nesse trabalho, é a segurança da informação, através da utilização do protocolo HTTPS, que permite troca de informações criptografadas.

As propriedades e atributos, são as informações e funções que determinado dispositivo gerenciado possui. Essas informações e funções podem ser lidas ou acessadas a partir de instruções enviados da aplicação centralizadora. As



propriedades e atributos, serão disponibilizadas de forma dinâmica na aplicação centralizadora, com isso, o usuário não necessita de alterar o *layout* cada vez que há alterações de propriedades e atributos.

No capítulo 3, será apresentada a metodologia utilizado para o desenvolvimento dos protótipos, bem como o processo de validação dos sistemas.

### **3 METODOLOGIA**

A metodologia científica envolve hipótese teórica, que por sua vez encaminha a um referencial teórico. A ciência estudada os procedimentos de pesquisa, remetem a relatórios e a coleta de informações, realizam a análise e interpretação de dados (JUNIOR, 2011).

Neste trabalho, será realizado um estudo das tecnologias que abrange toda a área macro de IoT, através da pesquisa de referencial teórico e estudo de trabalhos relacionados. Em um segundo momento, será realizado um experimento prático, verificando a possibilidade dos dispositivos suportarem as tecnologias abordadas neste trabalho.

Após a experimentação tecnológica, será desenvolvido um protótipo para demonstrar o funcionamento da ideia proposta, permitindo realizar uma comunicação entre a aplicação centralizadora e aplicação gerenciada.

#### **3.1 Procedimentos de pesquisa**

Neste trabalho será empregada uma metodologia de pesquisa experimental, que conforme Gerhardt e Silveira (2009), busca visar o estudo, através do desenvolvimento de protótipos, capazes assim de influenciar em uma determinada forma as implicações do protótipo.

Os próximos capítulos, estão enumerados pelos procedimentos de pesquisas, onde apresentam-se o referencial teórico, desenvolvimento dos protótipos e pôr fim, a validação do protótipo desenvolvido.

### **3.2 Referencial Teórico**

Este trabalho tem como objetivo de conhecer, compreender e buscar referenciais e embasamentos teóricos apresentados pelos principais autores das áreas de IoT, software e hardware.

Conforme Junior, 2011, não existe uma metodologia específica e sim uma opção adequada para cada caso, porém todas as situações que envolvem o problema precisam ser estudadas.

Ao propor a criação do protótipo da aplicação centralizadora e da aplicação gerenciada, buscou-se trabalhos relacionados, que pudessem contribuir para o desenvolvimento deste trabalho.

### **3.3 Desenvolvimento dos protótipos**

No capítulo 2, foi apresentado o referencial teórico das tecnologias de hardware, software e protocolos de comunicação que serão utilizados para o desenvolvimento dos protótipos.

Durante o desenvolvimento dos protótipos, apresentados no capítulo 4, será utilizado a metodologia de desenvolvimento evolucionário. O desenvolvimento evolucionário, segundo Pressman, 2011, consiste em produzir um modelo inicial e refiná-lo ao longo das interações das várias fases de desenvolvimento de um software, consequentemente, atingir a forma final do software.

Para a elaboração desse trabalho, serão desenvolvidos dois protótipos para demonstrar a aplicação funcional. Será criado um *software* cliente, aplicação centralizadora e um *software* agente, aplicação gerenciada. O protocolo de comunicação será definido no capítulo 4.3.

### 3.4 Processo de validação da proposta

Para testar as aplicações, num primeiro momento, será instalado em dois locais distintos, duas Raspberry Pi 3, com o intuito de conectar-se através da aplicação centralizadora, com a finalidade de obter o *status* de cada equipamento.

No segundo momento, serão dispostos sensores nas GPIO de saída da Raspberry Pi 3, sensor de temperatura, sensor de pressão, sensor de luminosidade e sensor de umidade, por fim, um diodo emissor de luz.

O objetivo é implementar diversos algoritmos na aplicação gerenciada, com o objetivo de realizar a coleta de informações dispostas pelos sensores conectados na Raspberry Pi 3. Além da coleta das informações, será disposto a funcionalidade para acionar o diodo emissor de luz, através da aplicação centralizadora.

Por fim, na aplicação centralizadora, será validado a obtenção das informações de cada dispositivo, neste caso, será obtida dados como: temperatura ambiente, pressão, umidade, luminosidade e o *status* atual do diodo emissor de luz.

No próximo capítulo, será demonstrado o desenvolvimento de um protótipo para controle de dispositivos distribuídos, será detalhado a utilização dos protocolos de comunicação e de bibliotecas que facilitam o desenvolvimento de software.

## 4 DESENVOLVIMENTO

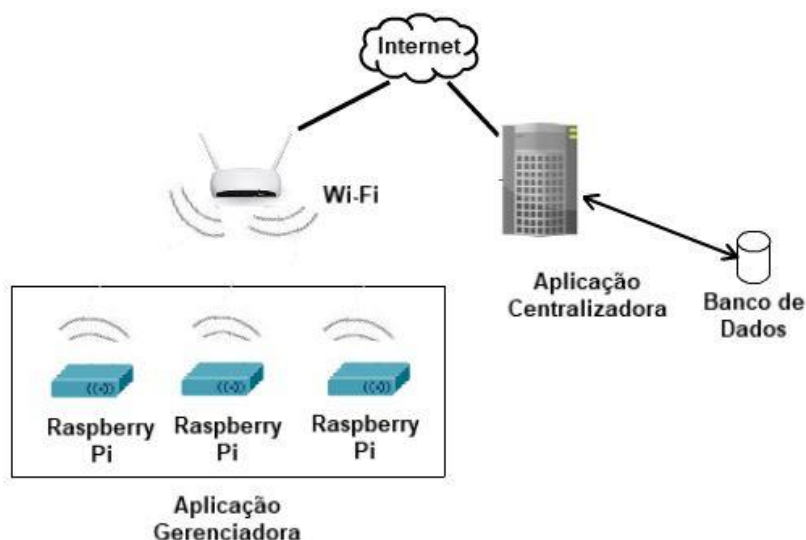
No capítulo 2 foram apresentadas as plataformas de hardware e de software, inclusive os protocolos de interoperabilidade, que compõem este trabalho. Neste capítulo serão apresentadas, a implementação de um protótipo de aplicação centralizadora e a aplicação gerenciada.

Foi desenvolvido um exemplo dos protótipos como prova de conceito, segundo Pinheiro (2010), a prova de conceito tem como objetivo validar um projeto antes que este seja executado na prática. Com esse propósito, será demonstrado a funcionalidade dos protótipos da aplicação centralizadora e da aplicação gerenciada nas seções 4.1 e 4.2.

Neste trabalho é proposto um protótipo de sistema composto por duas partes, a parte centralizadora e a parte de gerenciamento dos dispositivos distribuídos da IoT. A aplicação centralizadora será hospedada em um servidor com a aplicação Tomcat, sua especificação pode ser conferida na seção 2.2.2, juntamente com o banco de dados PostgreSQL, apresentado na seção 2.2.4. O servidor está conectado à Internet e a rede local, de forma que os dispositivos da IoT possam alcançar o mesmo.

A aplicação gerenciada está implementada em um Raspberry Pi 3, que também executa uma instância do servidor de aplicação Tomcat. A aplicação gerenciada está sempre aguardando um comando, a ser enviado pela aplicação centralizadora, para executar uma ação ou retornar uma informação relevante. Na Figura 11, é apresentado um diagrama macro da visão do projeto, nesta figura é apresentado a arquitetura do sistema e sua operabilidade.

Figura 11 – Arquitetura do Sistema



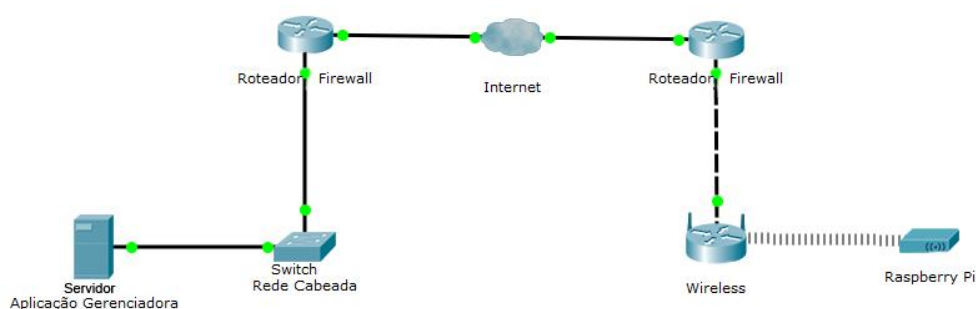
Fonte: Autor, 2017

A implementação do software foi realizada utilizando a linguagem de Programação Java. A principal característica para a escolha dessa linguagem é a portabilidade fornecida pela linguagem, que permite implementar aplicações multiplataforma, seja para *Web*, *desktop* ou *mobile*, através da execução da JVM, apresentada na seção 2.2.1. Outra motivação para a escolha da linguagem, deve-se a disponibilidade de *frameworks* e bibliotecas, que auxiliam no desenvolvimento de software, abstraindo a complexidade da implementação.

Para complementar a implementação da aplicação centralizadora, foi utilizada a linguagem de programação JavaScript com JQuery, que possibilita a execução de *scripts* no navegador do cliente, tornando as páginas da *Web* com resposta mais dinâmica. O uso do Bootstrap permitiu elaborar design e estilos, reaproveitando componentes, tendo como objetivo a melhor programação visual, logo a aparência da aplicação.

O sistema utiliza a conexão de uma rede local até a Internet, conforme a arquitetura de rede, demonstrado na Figura 12.

Figura 12 – Arquitetura de rede do Protótipo



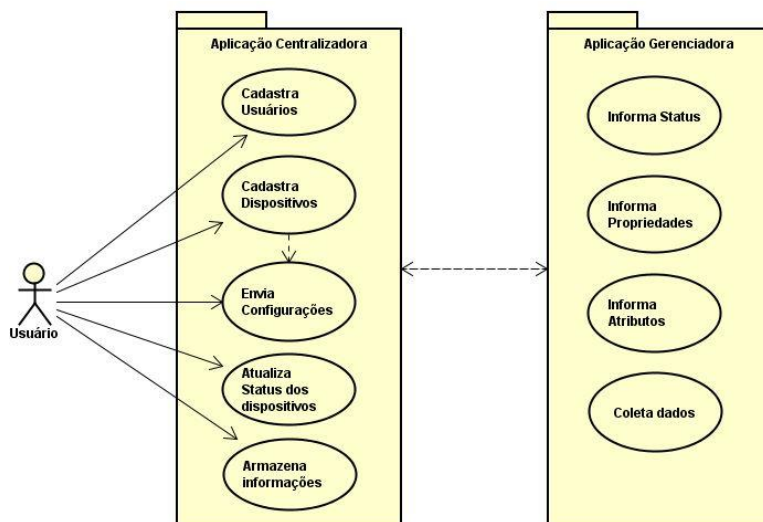
Fonte: Autor, 2017

Os dispositivos distribuídos poderão conectar-se a aplicação centralizadora, através dos protocolos de rede *Wi-Fi* ou estrutura cabeada, conforme especificado na seção 2.3.1.

Conforme Larman, 2000, os diagramas apresentam as principais interações que ocorrem entre os agentes e sistemas, permitindo associar atores ao caso de uso, limites de utilização do sistema e relacionamento entre os elementos. Desta maneira, foi elaborado um diagrama de caso de uso e um diagrama de sequência, para demonstrar as interações do agente com a aplicação centralizada e com a aplicação gerenciada.

A Figura 13, demonstra o diagrama de caso de uso, nela é apresentando o relacionamento entre o usuário, juntamente com suas atribuições, possibilitando ao usuário cadastrar novos usuários da aplicação, dispositivos, enviar configurações para aplicação gerenciada e por fim, realizar sincronizações de informações. A aplicação gerenciada possui funcionalidades que consistem em enviar informações para a aplicação centralizadora, bem como informar propriedades e atributos.

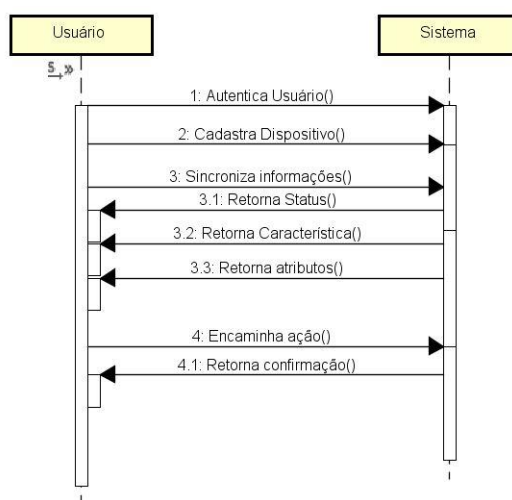
Figura 13 – Diagrama de Caso de Uso



Fonte: Autor, 2017

A Figura 14, demonstra o diagrama de sequência, visa representar as sequências dos processos com a interação do usuário com o protótipo desenvolvido. O diagrama demonstra as principais ações que o usuário pode efetuar, a sequência de cada ação é apresentada um retorno do sistema, no qual fica encarregado de demonstrar o resultado da requisição.

Figura 14 – Diagrama de Sequência



Fonte: Autor, 2017

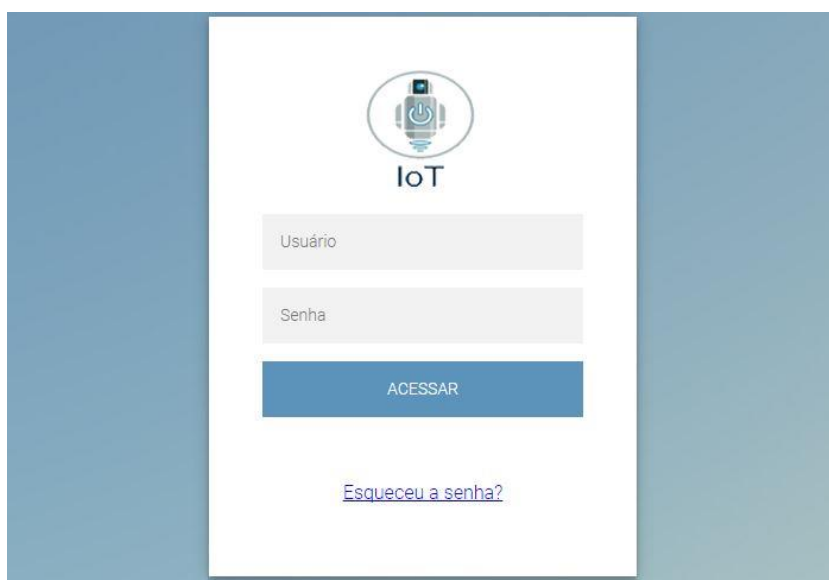


Nas próximas seções serão apresentadas informações detalhadas da aplicação centralizadora, aplicação gerenciada e a validação dos protótipos.

#### 4.1 Aplicação Centralizadora

A aplicação centralizadora, será desenvolvida em uma interface *Web*. A interface será desenvolvida com JavaScript, JQuery e por fim, a interface visual e seus componentes com o *framework* Bootstrap em HTML. Na Figura 15 é apresentado a interface de acesso a aplicação centralizadora.

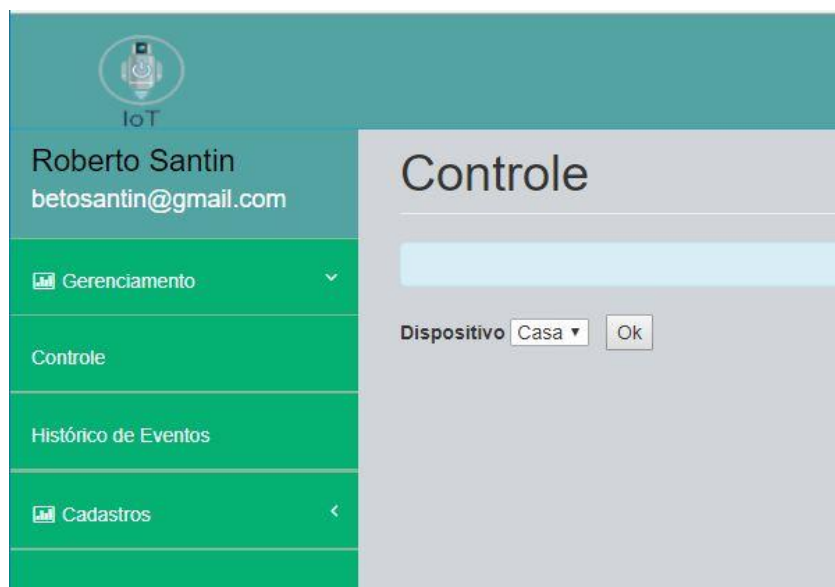
Figura 15 – Interface de acesso a aplicação centralizadora



Fonte: Autor, 2017

Ao realizar o acesso a aplicação centralizadora, é disponibilizado ao usuário um menu lateral, como é possível visualizar na Figura 16, na opção de Controle, é possível selecionar o dispositivo a ser controlado.

Figura 16 – Menu de opções



Fonte: Autor, 2017

Após seleciona o dispositivo a ser controlado, o sistema cria a estrutura dinâmica, apresentando um formulário para executar determinadas ações no dispositivo gerenciado.

A estrutura dinâmica é criada a partir de informações das propriedades e atributos recebidos a partir da sincronização de dados com o dispositivo gerenciado, realizado no momento em que o dispositivo gerenciado é cadastrado.

Na Figura 17 é demonstrado o formulário dinâmico criado pela aplicação centralizadora.

Figura 17 – Formulário dinâmico

The screenshot shows a web application interface for IoT device control. On the left is a green sidebar with a user profile at the top: 'Roberto Santin' and 'betosantin@gmail.com'. Below the profile are menu items: 'Gerenciamento' (with a dropdown arrow), 'Controle', 'Histórico de Eventos', and 'Cadastros' (with a left arrow). The main content area is titled 'Controle' and features a light blue header bar. Below this, there is a 'Dispositivo' dropdown menu set to 'Casa' with an 'Ok' button. The 'Função' is set to 'acenderled'. There are five parameter input fields, each with a label and a text box: 'Parâmetro: red' (value: red), 'Parâmetro: green' (value: green), 'Parâmetro: blue' (value: blue), 'Parâmetro: time' (value: time), and 'Parâmetro: blink' (value: blink). At the bottom of the form is a blue 'Executar' button.

Fonte: Autor, 2017

O objetivo do formulário dinâmico, é tornar a aplicação centralizadora totalmente apta a receber outros dispositivos, sem realizar modificações na aplicação.

A aplicação centralizadora, dispõe de uma tabela que armazena todos os eventos enviados, pode ser encontrado na opção Histórico de Eventos. Na Figura 18 pode ser visualizado informações coletadas para a aplicação gerenciada.

Figura 18 – Histórico de Eventos

**Histórico de Eventos**

Tabela de eventos

10 records per page Search:

Nome Dispositivo	Usuário	Evento	Quando	Direção	Valor
Casa	Administrador	acenderled	10/10/2017 20:15:22	Enviado	255,255,255
Casa	Administrador	acenderled	10/10/2017 20:15:24	Retorno	Ok
Casa	Administrador	obtertemperatura	11/10/2017 22:07:33	Enviado	
Casa	Administrador	obtertemperatura	11/10/2017 22:07:36	Retorno	28°C

Showing 1 to 4 of 4 entries

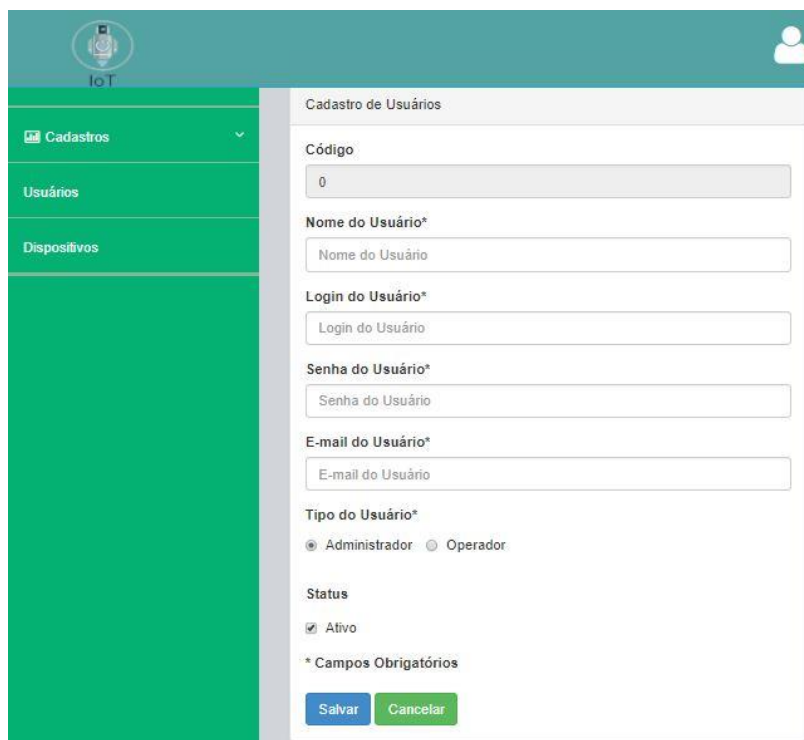
Previous 1 Next

Fonte: Autor, 2017

A Figura 19 apresenta a opção de Cadastros de usuários, nela o sistema possibilita realizar a consulta dos usuários já cadastrados, bem como realizar o cadastro de novos usuários, esses usuários podem ter acesso a aplicação centralizadora.

O cadastro de usuários é composto pelas seguintes informações: Nome do Usuário, *Login*, Senha, E-mail, Tipo do Usuário e por fim o seu *Status* atual, todas as informações são de preenchimento obrigatório.

Figura 19 – Cadastro de Usuário



The screenshot displays a web application interface for user registration. On the left, a green sidebar contains a menu with 'Cadastros' (expanded), 'Usuários', and 'Dispositivos'. The main area is titled 'Cadastro de Usuários' and contains the following fields: 'Código' (text box with '0'), 'Nome do Usuário\*' (text box with placeholder 'Nome do Usuário'), 'Login do Usuário\*' (text box with placeholder 'Login do Usuário'), 'Senha do Usuário\*' (text box with placeholder 'Senha do Usuário'), 'E-mail do Usuário\*' (text box with placeholder 'E-mail do Usuário'), and 'Tipo do Usuário\*' (radio buttons for 'Administrador' and 'Operador'). Below these is a 'Status' section with a checked 'Ativo' checkbox. A note '\* Campos Obrigatórios' is present above 'Salvar' and 'Cancelar' buttons.

Fonte: Autor, 2017

Na Figura 20 é demonstrado o cadastro de dispositivos, possibilitando consultar e cadastrar dispositivos IoT. O cadastro é composto das principais informações dos dispositivos, são elas: Nome do Dispositivo, Endereço de IP, Porta de Conexão, Serviço e a *Servlet*, todos os campos são de preenchimento obrigatório.

O cadastro dos dispositivos é obrigatório, pois através das informações preenchidas serão feitas as sincronizações, conexões e troca de informações.

Figura 20 – Cadastro de Dispositivos

The image shows a web application interface for IoT device management. On the left is a green sidebar with a menu containing 'Cadastrados', 'Usuários', and 'Dispositivos'. The main area is titled 'Cadastro de Usuários' and contains a form with the following fields: 'Código' (with value '0'), 'Nome do Dispositivo\*' (placeholder 'Nome do Dispositivo'), 'Endereço de IP\* Ex: https://192.168.2.2' (placeholder 'Endereço de IP'), 'Porta\* Ex: 8443' (with value '80'), 'Serviço\*' (placeholder 'Serviço'), and 'Servlet\*' (placeholder 'Servlet'). Below the form is a note '\* Campos Obrigatórios' and two buttons: 'Salvar' (blue) and 'Cancelar' (green).

Fonte: Autor, 2017

As informações cadastradas na aplicação centralizadora ou coletas nos dispositivos gerenciados, estão armazenadas no banco de dados. Ele é composto por cinco tabelas relacionais, nessas tabelas, estão armazenadas informações de Usuários, Históricos, Dispositivos, Métodos e Parâmetros.

A tabela de Usuário, armazena informações essenciais como: Nome, *Login*, Senha, Perfil do Usuário, E-mail e *Status*.

A tabela de Históricos, armazena as informações recebidas dos dispositivos gerenciados, dispostas para o usuário na aplicação centralizadora. Os dados armazenados são: Dispositivo Gerenciado, Usuário do dispositivo, Evento disparado, Data e Hora do ocorrido, Direção e por fim, o Valor coletado.

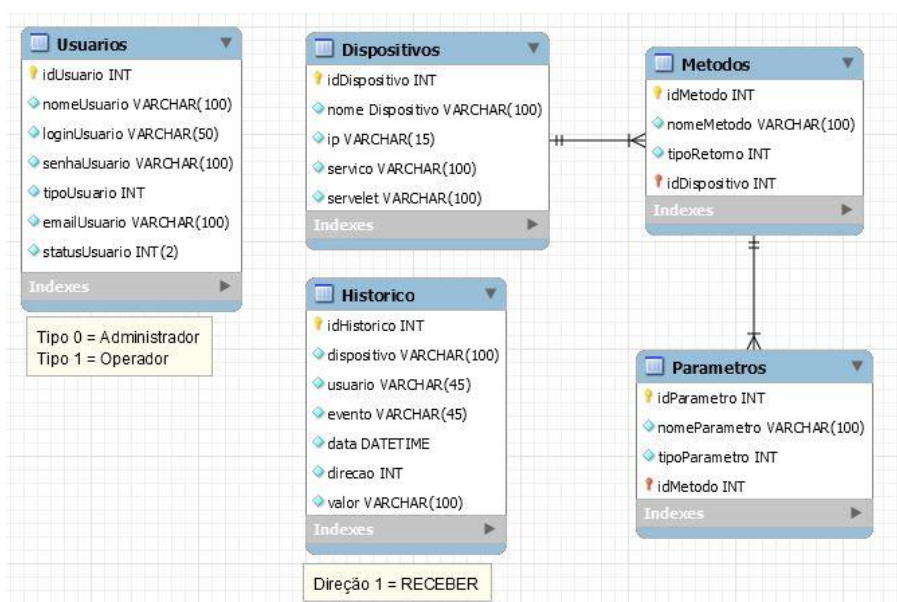
A tabela de Dispositivos, reúne as informações cadastrais referente aos dispositivos gerenciados, nela são armazenadas informações como: Nome do Dispositivo, Endereço de IP Válido, Serviço e a *Servlet*.

A tabela de Métodos, são armazenadas as informações da aplicação gerenciada, estão armazenadas informações como: Nome do Método, Tipo do Retorno, Vinculação com o dispositivo.

A tabela de Parâmetros, são armazenadas as informações referente aos parâmetros dos métodos, contendo informações como: Nome do Parâmetro, Tipo do valor do Parâmetro e por fim, a vinculação com o método o qual pertence.

A Figura 21, demonstra o modelo relacional utilizado na aplicação centralizadora.

Figura 21 – Modelo relacional do banco de dados



Fonte: Autor, 2017

As tabelas apresentadas, são de suma importância para o bom funcionamento da aplicação centralizadora, bem como para o funcionamento eficaz da aplicação gerenciada.

As tabelas de Métodos e Parâmetros, contém informações para criar a estrutura de formulário dinâmico, conforme apresentado na Figura 17.

Na próxima seção será demonstrado o funcionamento da aplicação gerenciada, apresentando um exemplo de acionamento de um diodo emissor de luz e a leitura de dados dos sensores de temperatura, pressão e umidade.

## 4.2 Aplicação Gerenciada

A aplicação gerenciada utilizará a plataforma de hardware Raspberry Pi 3, conforme especificações apresentadas na seção 2.1.1, esta plataforma utiliza um servidor *Web Tomcat*, conforme especificado na seção 2.2.2. A aplicação gerenciada, permanece aguardando por um comando. O comando pode ser para coletar informações ou executar uma determinada ação definida pela aplicação centralizadora.

A aplicação gerenciada deve obedecer ao protocolo estabelecido no capítulo 4.3, independente da linguagem de programação desenvolvida ou do dispositivo utilizado para gerenciar sensores e demais equipamentos conectados nas portas externas do dispositivo.

Neste trabalho, foi utilizado a plataforma Raspberry Pi 3, nela foi instalada a aplicação gerenciada, assim pode-se realizar o controle diretamente na aplicação centralizada, conforme o objetivo desse trabalho.

Para realizar a ativação do dispositivo gerenciado, foram utilizados alguns componentes, são eles: LED tipo RGB e o Sensor BME 280.

O diodo emissor de luz, LED, permite ser habilitado com a combinação das três cores primárias, vermelho, verde e azul. Ao acionar o diodo a partir do formulário dinâmico localizado na aplicação centralizadora, a aplicação receberá os parâmetros de cores e em seguida será encaminhado o comando para acionar o LED, através das saídas GPIO da Raspberry PI 3.

O sensor BME 280, é composto pelos sensores de medição de temperatura, pressão e umidade. O BME 280, é interligado através do barramento I2C<sup>5</sup> da Raspberry Pi 3, a partir desse barramento, é possível enviar e receber dados dos componentes conectados. O sistema coletará os dados de temperatura, humidade e pressão.

---

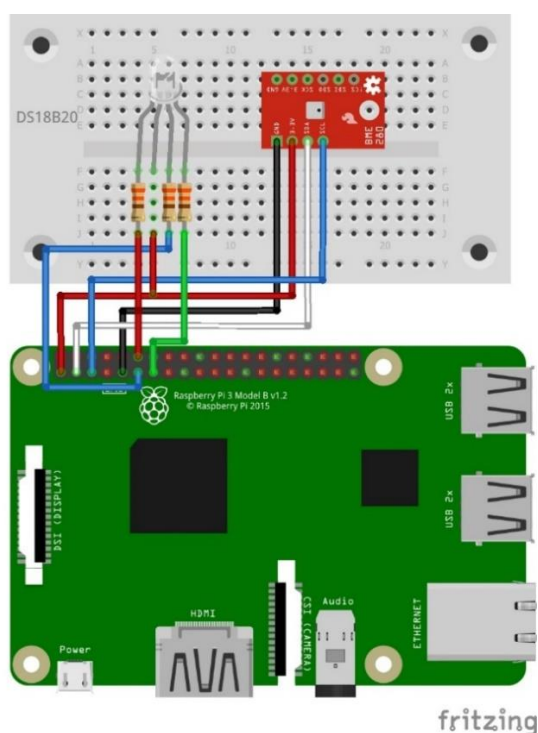
<sup>5</sup> I2C, *Inter-Integrated Circuit*, é um barramento serial bidirecional, utilizado para conectar periféricos de baixa velocidade a uma placa central ou em um sistema embarcado.



A Figura 22 demonstra o diagrama de conexão entre os componentes. O diodo emissor de luz, necessita receber quatro conexões, no exemplo proposto, foi realizada a conexão da alimentação, Vcc e três conexões tipo PWM.

O sensor BME 280 é conectado através das conexões Vcc, GND e conexões para o barramento I2C.

Figura 22 – Diagrama ligação diodo emissor de luz e sensor BME 280



Fonte: Autor, 2017

No Código 8 é apresentado um trecho do código fonte criado para realizar o acionamento do diodo emissor de luz. Na aplicação gerenciada foi implementado mecanismos de *threads* para permitir que a aplicação não fique bloqueada até a conclusão da atividade designada.

### Código 7 – Acionamento do diodo emissor de luz

```
public void init( int red, int green, int blue ) {
    Gpio.wiringPiSetup();
    // Criadas as portas PWM para cada pino com a intensidade de 0 a 100
    SoftPwm.softPwmCreate(pinLayout.getRedPin().getAddress(), 0, 100);
    SoftPwm.softPwmCreate(pinLayout.getGreenPin().getAddress(), 0, 100);
    SoftPwm.softPwmCreate(pinLayout.getBluePin().getAddress(), 0, 100);

    // Escrita na porta PWM, convertendo o valor de cada variável na base de 0 a 100
    SoftPwm.softPwmWrite(pinLayout.getRedPin().getAddress(), convert( red, 0, 100) );
    SoftPwm.softPwmWrite(pinLayout.getGreenPin().getAddress(), convert( green, 0,100) );
    SoftPwm.softPwmWrite(pinLayout.getBluePin().getAddress(), convert( blue, 0,100) );
}
```

Fonte: Autor, 2017

No Código 9 é demonstrado uma abstração do código fonte criado para realizar o obter os dados disponibilizados pelo sensor BME 280.

Na aplicação gerenciada foi implementado mecanismos para enviar comandos através do barramento I2C e realizar as leituras dos valores retornado pelo sensor.

### Código 8 – Captura de dados do Sensor BME 280

```
public float obterPressao() {
    return new bme280().readPressure();
}

public float obterTemperatura() {
    return new bme280().readTemperature();
}

public float obterUmidade() {
    return new bme280().readHumidity();
}
```

Fonte: Autor, 2017

Na próxima seção será demonstrado o desenvolvimento do protocolo de comunicação, interligando a aplicação centralizadora e a aplicação gerenciada.

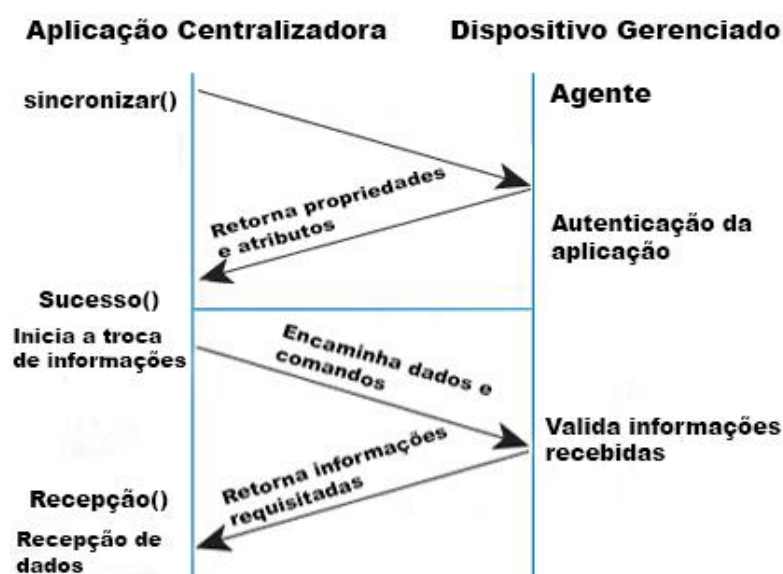
### 4.3 Protocolo

A construção do protocolo utilizado neste trabalho, é baseado em protocolos de comunicação e interoperabilidade já existentes, conforme apresentado no capítulo 2.3.

O protocolo possui o processo de troca de mensagens, que conforme Comer (2007), o aperto de mão, *handshake*, é a parte mais importante da comunicação entre dois dispositivos autônomos, dessa forma, há uma definição para transmissão e recepção de informações entre os dispositivos.

Na Figura 23 pode-se observar o processo de troca de mensagens, neste processo é demonstrado a sincronização, autenticação e troca de informações entre o dispositivo e a aplicação.

Figura 23 – Processo de troca de mensagens



Após realizar o cadastro do dispositivo gerenciado, o agente, é necessário realizar a primeira sincronização de dados. Nesta etapa, a aplicação centralizadora recebe as propriedades e atributos que o dispositivo gerenciado possui, além de realizar a troca de informações de segurança da informação, na qual uma chave criptografada é transferida, necessária para validar os dados recebidos e enviados. Essas informações são relevantes para montar a estrutura de ações dinâmicas, conforme demonstrado na Figura 17.

O Código 9 demonstra uma abstração do código fonte que realiza o processo de sincronização do dispositivo gerenciado com a aplicação centralizadora.

#### Código 9 – Exemplo da sincronização com o protocolo

```
public void sincronizarDispositivo( int id ) throws IOException {
    Dispositivo d = ds.getSource(id);
    if ( d != null ) {
        String urlString = d + "sincronizar";
        BufferedReader rd = null;

        try {
            URL url = new URL(urlString);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("POST");
            conn.setReadTimeout(10000);
            rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            TypeToken<List<Metodos>> token = new TypeToken<List<Metodos>>() {
            };
            List<Metodos> personList = new Gson().fromJson(rd.readLine(), token.getType());
            rd.close();
        } catch (IOException ex) {
            requisicao.getSession().setAttribute("retorno", "Timeout, verifique o endereço IP, Porta,
            Serviço e Servelet da conexão");
            ex.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Fonte: Autor, 2017

A Figura 24 demonstra o retorno das propriedades e atributos, via JSON, que o dispositivo gerenciado retorna quando realizado a sincronização inicial.

Figura 24 – Retorno das propriedades e atributos

```
[{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"acenderled","tipoRetorno":0,"parametros":
[{"idParametro":0,"idMetodo":0,"nomeParametro":"red","tipo":2},
{"idParametro":0,"idMetodo":0,"nomeParametro":"green","tipo":2},
{"idParametro":0,"idMetodo":0,"nomeParametro":"blue","tipo":2},
{"idParametro":0,"idMetodo":0,"nomeParametro":"time","tipo":2},
{"idParametro":0,"idMetodo":0,"nomeParametro":"blink","tipo":3}]],
{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"apagarled","tipoRetorno":0},
{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"obtertemperatura","tipoRetorno":4},
{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"obterpressao","tipoRetorno":4},
{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"obterhumidade","tipoRetorno":4},
{"idMetodo":0,"idDispositivo":0,"nomeMetodo":"obterldr","tipoRetorno":2}]
```

Fonte: Autor, 2017

Após receber as informações de propriedades e atributos, a aplicação centralizadora, já possui condições de realizar a comunicação com o dispositivo gerenciado.

O usuário através da aplicação centralizadora, realiza o envio de configurações ou ações que a aplicação gerenciada deve executar, o histórico de eventos recebidos, são registrados na tabela de históricos, conforme demonstrado na Figura 18.

A aplicação centralizadora, cria a estrutura do formulário em HTML dinamicamente, conforme propriedades e atributos obtidos da aplicação gerenciadora, o Código 10 demonstra um fragmento do formulário criado dinamicamente.

## Código 10 – Formulário dinâmico

```
<form name="acenderled" action="https://192.168.2.2:8443/Agente/acao?acenderled"
method="post">
  <div class="form-group">
    <label>Parâmetro: red</label>
    <input name="red" class="form-control"
placeholder="red" type="number"></div>
  <br>
  <div class="form-group">
    <label>Parâmetro: green</label>
    <input name="green" class="form-control"
placeholder="green" type="number"></div>
  <br>
  <div class="form-group">
    <label>Parâmetro: blue</label>
    <input name="blue" class="form-control"
placeholder="blue" type="number"></div>
  <br>
  <div class="form-group">
    <label>Parâmetro: time</label>
    <input name="time" class="form-control"
placeholder="time" type="number"></div>
  <br>
  <div class="form-group">
    <label>Parâmetro: blink</label>
    <input name="blink" class="form-control"
placeholder="blink" type="text"></div>
  <br><button type="submit" class="btn btn-primary">Executar</button>
</form>
```

Fonte: Autor, 2017

A estrutura criada, trata-se de um formulário com o método POST, ou seja, este tipo de método HTTP, submete dados para serem processados pelo recurso encontrado no dispositivo gerenciado, registrando as informações enviadas na tabela de históricos de eventos.

Por fim, após realizar o envio da ação para o dispositivo gerenciado, o sistema aguardará um retorno do dispositivo gerenciado. O histórico de eventos demonstra as atividades realizadas.

Para cada nova requisição realizada pela aplicação centralizada, a aplicação gerenciada responde com uma nova conexão, conectando-se a aplicação centralizadora para salvar os dados encaminhados. Assim pode ser evitado problemas em que a aplicação centralizadora fique bloqueada, aguardando um retorno da aplicação gerenciada.

Foi possível evidenciar uma restrição de uso, o funcionamento desse protocolo restringe-se a utilizar a aplicação centralizadora e o dispositivo gerenciado com IP verdadeiro.

Na próxima seção será demonstrado a importância da segurança da informação, bem como o processo de desenvolvimento de métodos de segurança nos protótipos desenvolvidos.

#### **4.4 Segurança da Informação**

Conforme a seção 2.3.2, a segurança da informação neste trabalho, compreende em utilizar certificados digitais, para criptografar as trocas de mensagens entre dispositivos, através do protocolo HTTPS, ou seja, evitar o roubo de informações através de ataques cibernéticos.

Para o desenvolvimento do trabalho, foi criado um certificado *self-signed*, ou seja, sem a certificação de um órgão emissor. De acordo com a Global Sign (2017) e Monteiro (2007), a certificação e emissão do certificado por um órgão emissor, atesta a veracidade e confiabilidade da transferência dos dados e informação criptografadas, evitando assim, o roubo de informações confidenciais. A Global Sign (2017), alerta para os perigos ao utilizar um certificado não validado por uma autoridade certificadora.

Ciente disso, a implementação dos protótipos utilizando certificado auto assinado somente para fim de prova de conceito, pois Monteiro (2007), recomenda a

utilização de um certificado válido, para maximizar a segurança e privacidade dos dados.

O certificado em questão, foi criado através da criptografia RSA, essa criptografia é considerada a mais segura do mundo, conforme Monteiro, 2007, não há tentativas sucedidas de quebra de criptografia. O algoritmo RSA, envolve um par de chaves, a chave pública, que pode ser conhecida por todos e a chave privada, na qual deve ser mantida em segredo. Todas as mensagens criptografadas utilizando a chave pública, só podem ser descriptografadas utilizando a chave privada, na qual, só o proprietário a conhece.

Para criar o certificado auto assinado, utilizamos o *keytool*, disponibilizado através da biblioteca *Java Development Kit* (JDK). No Sistema Operacional Windows, necessitamos executar em modo de administrador do sistema operacional, o *Command Prompt* (CMD).

A Figura 25 apresenta o processo de emissão do certificado auto assinado. No preenchimento do formulário, deve ser informado os seguintes dados: Senha para criptografia, nome completo, cidade, estado e país, ao fim deste processo, será criado o certificado auto assinado com o nome de “MeuCertificado.cert”.

Figura 25 – Emissão do Certificado auto assinado

```

Administrador: Prompt de Comando
C:\Arquivos de Programas\Java\jdk1.8.0_92\bin>keytool.exe -genkeypair -alias MeuCertificado -keyalg RSA -keystore "c:\MeuCertificado.cert"

Informe a senha da área de armazenamento de chaves:
Informe novamente a nova senha:
Qual é o seu nome e o seu sobrenome?
[Unknown]: ROBERTO SANTIN
Qual é o nome da sua unidade organizacional?
[Unknown]: ROBERTOSANTIN.COM.BR
Qual é o nome da sua empresa?
[Unknown]: RS
Qual é o nome da sua Cidade ou Localidade?
[Unknown]: ENCANTADO
Qual é o nome do seu Estado ou Município?
[Unknown]: RIO GRANDE DO SUL
Quais são as duas letras do código do país desta unidade?
[Unknown]: BR
CN=ROBERTO SANTIN, OU=ROBERTOSANTIN.COM.BR, O=RS, L=ENCANTADO, ST=RIO GRANDE DO SUL, C=BR Está correto?
[não]: SIM

Informar a senha da chave de <MeuCertificado>
(RETURN se for igual à senha da área do armazenamento de chaves):
Informe novamente a nova senha:
C:\Arquivos de Programas\Java\jdk1.8.0_92\bin>

```

Fonte: Autor, 2017

O processo de emissão do certificado em outros Sistemas Operacionais, segue o mesmo processo, porém em ferramentas específicas para cada Sistema Operacional.



Após a emissão, precisamos configurar o servidor Apache Tomcat, para utilizar o certificado criado, além de utilizar o protocolo HTTPS, assim, as conexões são criptografadas com o certificado criado, através do algoritmo RSA.

O Código 11 apresenta a alteração necessária nas configurações do Apache Tomcat, localizado no arquivo server.xml, a chave privada está identificada pelo parâmetro “keystorePass”.

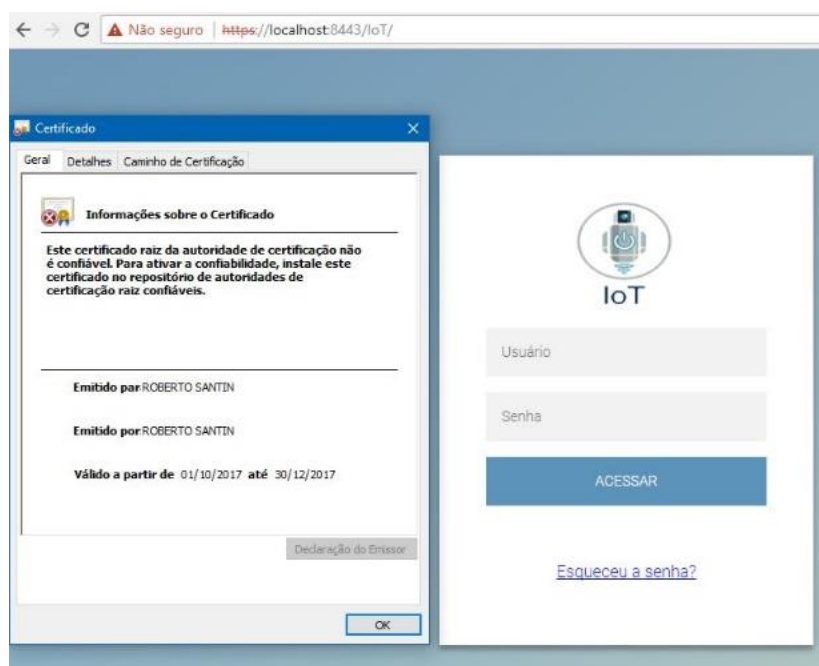
#### Código 11 – Configuração Apache Tomcat

```
<Connector SSLEnabled="true" clientAuth="false"
    keystoreFile="C:/MeuCertificado.cer" keystorePass="robertosantin tcc iot"
    maxThreads="150" port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    scheme="https" secure="true" sslProtocol="TLS"/>
```

Fonte: Autor, 2017

A Figura 26 apresenta a utilização do certificado ao acessar a aplicação centralizadora, conforme descrito no início do capítulo.

Figura 26 – Certificado auto assinado



Fonte: Autor, 2017

Na próxima seção será demonstrado o processo de validação dos protótipos desenvolvidos nesse trabalho.

#### **4.5 Processo de Validação**

O processo de validação, foi iniciado a partir da montagem dos componentes conectando nas conexões de saídas da Raspberry Pi 3, conforme é demonstrado na seção 4.2, Figura 22.

Duas Raspberry Pi 3, foram instaladas em dois locais distintos, com o intuito de conectar-se através da rede local na aplicação centralizadora, assim os dispositivos gerenciados foram testados para enviar e obter informações dos componentes instalados nas conexões de saída da Raspberry Pi 3.

As informações obtidas foram adequadamente salvas no histórico de eventos, permitindo analisar os dados de temperatura, pressão e humidade, logo após realizar o envio da requisição solicitando as informações. Como prova de conceito, foram enviadas configurações e realizado o acionamento do diodo emissor de luz, ambas as situações foram feitas com sucesso, devidamente registradas no histórico de eventos.

A partir da estrutura demonstrada na Figura 11, através da internet, foi possível realizado troca de informações conforme foi descrito no parágrafo anterior. A restrição de uso apresentada na seção 4.3, em que ambas as aplicações devem possuir IP verdadeiro, não causou problemas de desempenho ou funcionalidades prejudicadas.

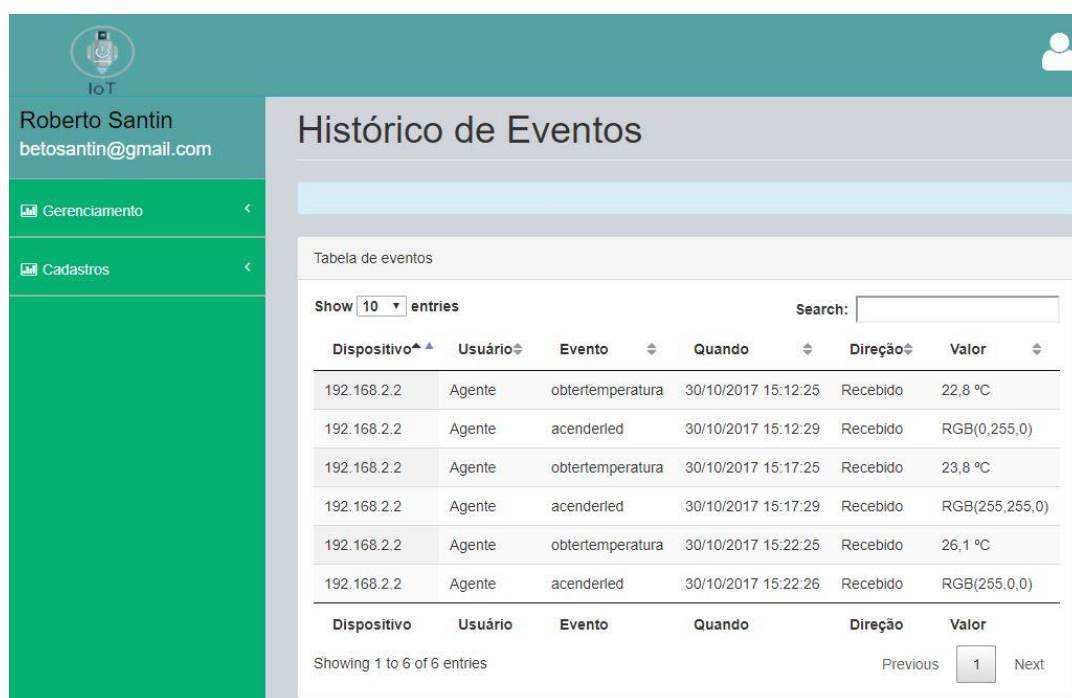
No segundo processo de validação, foi realizado um monitoramento da temperatura ambiente de um *datacenter*. A cada cinco minutos, foi realizado uma leitura da temperatura ambiente.

A cada leitura, é realizado uma verificação, seguindo a seguinte regra: Entre a temperatura de 20° C e 23° C, foi encaminhado o acionamento do LED na cor verde. Entre a temperatura de 23° C até 25° C, foi encaminhado o acionamento do LED na cor amarelo. Quando a temperatura for menor que 20° C ou maior que 25° C, foi encaminhado o acionamento do LED na cor vermelha.

O sistema encaminhou os eventos diretamente para a aplicação centralizadora, registrando a temperatura aferida e a ação a ser tomada, ou seja, acionando o LED em determinadas cores previstas nas regras programadas na aplicação gerenciada.

Na Figura 27 é demonstrado os históricos de eventos encaminhados pela aplicação gerenciada para a aplicação centralizadora.

Figura 27 – Histórico de eventos recebidos no monitoramento do *Datacenter*



**Histórico de Eventos**

Tabela de eventos

Show 10 entries Search:

Dispositivo	Usuário	Evento	Quando	Direção	Valor
192.168.2.2	Agente	obtertemperatura	30/10/2017 15:12:25	Recebido	22,8 °C
192.168.2.2	Agente	acenderled	30/10/2017 15:12:29	Recebido	RGB(0,255,0)
192.168.2.2	Agente	obtertemperatura	30/10/2017 15:17:25	Recebido	23,8 °C
192.168.2.2	Agente	acenderled	30/10/2017 15:17:29	Recebido	RGB(255,255,0)
192.168.2.2	Agente	obtertemperatura	30/10/2017 15:22:25	Recebido	26,1 °C
192.168.2.2	Agente	acenderled	30/10/2017 15:22:26	Recebido	RGB(255,0,0)

Showing 1 to 6 of 6 entries Previous 1 Next

Fonte: Autor, 2017

Os eventos e informações enviadas pela aplicação gerenciada, foram salvas devidamente na tabela, processados pela aplicação centralizadora, sendo possível averiguar e manter um histórico das informações coletas.

No Capítulo 5 serão apresentadas as considerações finais em relação ao trabalho apresentado, além dos trabalhos futuros que podem ser utilizados com base nos protótipos desenvolvidos.

## 5 CONCLUSÃO

Neste trabalho foi proposto desenvolver uma aplicação centralizadora, que permite obter e realizar configurações em dispositivos distribuídos (aplicação gerenciada). As aplicações são conectadas através do protocolo desenvolvido no trabalho.

Ao buscar métodos e soluções para centralizar dados e informações obtidos a partir de um dispositivo IoT, foi proposta uma solução customizada, que através da automatização, busca alcançar facilidades como, centralização de controle e de informações coletadas, visando a proatividade no monitoramento e controle de dispositivos e componentes da IoT. Desta maneira, foi possível desenvolver um protótipo que atenda as demandas de centralizar o controle e coleta de informações dos dispositivos gerenciados.

Neste processo de iteração entre aplicação centralizadora e dispositivo distribuídos (IPs válidos), a partir da troca de informações através de eventos, foi possível validar e monitorar as ações executadas no dispositivo distribuído, tendo assim, flexibilidade para controlar os dispositivos de forma centralizada.

Devido ao autor não conhecer e/ou ter acesso a todas plataformas de hardware que poderão ser utilizados como dispositivos distribuídos, foi desenvolvido na aplicação centralizadora um mecanismo de formulários dinâmicos, que visa atender de forma satisfatória, o controle dos dispositivos distribuídos, podendo-se adequar as várias possibilidades de uso, restrito apenas aos tipos de dados reconhecidos pela aplicação, que podem ser extensíveis.

A aplicação centralizadora está publicada no GitHub<sup>6</sup>. Foi desenvolvida para possibilitar a interação entre os usuários e os dispositivos distribuídos, permitindo realizar o cadastro de usuários, dispositivos, que por sua vez, são necessários para realizar as operações. A aplicação centralizadora, dispõe da estrutura de formulário dinâmico, que permite executar comandos para coletar informações e envio de configurações.

A aplicação gerenciada (dispositivos distribuídos), utilizada para realizar a coleta de informações e recebimentos de comandos, encontra-se publicada no GitHub<sup>7</sup>, para o seu desenvolvimento, foi utilizado a biblioteca PI4J, que possibilitou a integração da aplicação gerenciada com os componentes conectados na Raspberry Pi 3. Foi possível abstrair a complexidade da implementação, mantendo o foco no desenvolvimento do protocolo proposto e execução das funções encaminhadas pela aplicação centralizadora.

No desenvolvimento do protocolo, foi implementado com base em outras estruturas e protocolos de comunicação já existentes. A utilização de JSON, HTTPS e técnicas M2M, proporcionaram condições para o desenvolvimento eficiente, facilitando as trocas de informações entre as aplicações. A partir disso, foi possível evidenciar uma restrição de uso, para o bom funcionamento do protocolo implementado, restringe-se a utilizar a aplicação centralizadora e o dispositivo gerenciado com IP verdadeiro.

## 5.1 Contribuições

Como principais contribuições deste trabalho, podemos destacar:

- Levantamento de soluções aplicados em produtos semelhantes;
- Desenvolvimento de uma solução baseada em centralização de dados e comandos;
- Flexibilização de controle dos dispositivos conectados
- Validação da API Pi4j;

---

<sup>6</sup> Códigos fonte da Aplicação Centralizadora: <https://github.com/betosantin/iot>

<sup>7</sup> Códigos fonte da Aplicação Gerenciada: <https://github.com/betosantin/Agente>

- Implementação de mecanismos para a segurança da informação;
- Interconexão da aplicação centralizadora e aplicação gerenciada;

## 5.2 Trabalhos futuros

Sugere-se para trabalhos futuros, utilizar esta plataforma em aplicações corporativas, integrando dispositivos na aplicação centralizadora, interconectando com sistemas de *Business Intelligence* (BI) e *Enterprise Resource Planning* (ERP).

Realizar comparativos com outras aplicações que utilizam uma sistemática semelhante a desenvolvida neste trabalho, permitindo assim, realizar testes de desempenho, avaliações de seguranças e avaliação de viabilidades técnica para outros casos a serem aplicados.

Implementar recursos para realizar execução de eventos a partir de agendamentos, permitindo autonomizar da aplicação centralizadora.

Criar mecanismos para suportar novos tipos de dados e novas validações, possibilitando assim, maximizar os componentes que podem ser atendidos, através de outras plataformas de *hardware*.

Propor novos meios de autenticação e segurança, realizar implementações para plataformas com menor poder de processamento, permitindo escolher o meio de segurança por dispositivo gerenciado, adaptando assim a estrutura da aplicação centralizadora.

Realização de testes de exaustão, visando atender ambientes de alta disponibilidade.

## REFERÊNCIAS

Apache Foundation. **Documentation of Apache Tomcat 9**. 2017. Disponível em:<  
<http://tomcat.apache.org/tomcat-9.0-doc/index.html>> Acesso em: 20 mar. 2017.

BARBOZA, Lucas C. **Modelo de arquitetura baseado em um Sistema de Internet das coisas aplicada a automação residencial**. São Carlos, 2015. Disponível em:<  
[http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-02022016-160158/publico/Barboza\\_Lucas\\_Carlos\\_tcc.pdf](http://www.tcc.sc.usp.br/tce/disponiveis/18/180500/tce-02022016-160158/publico/Barboza_Lucas_Carlos_tcc.pdf)> Acesso em: 30 mar. 2017.

BARROS, Marcelo. **MQTT - Protocolos para IoT**. 2015. Disponível em:<  
<https://www.embarcados.com.br/mqtt-protocolos-para-iot/>> Acesso em: 28 abr. 2017.

BAX, Marcello P. **Introdução às linguagens de marcas**. 2000. Disponível em:<  
<http://www.scielo.br/pdf/ci/v30n1/a05v30n1.pdf>> Acesso em: 01 mai. 2017.

Bleagle Board. **Introduction of the Bleaglebone**. 2017. Disponível em:<  
<https://beagleboard.org/Support/bone101>> Acesso em: 27 fev. 2017.

BOSWARTHICK, David. ELLOUMI, Omar. HERSENT, Olivier. **M2M communications: a systems approach**. John Wiley & Sons, Nova York, 2012.

BUTLER, J. et Al. **Wireless Networking in The Developing World**. Londres, 2013.

CHEE, J. S. Brian. FRANKLIN, **Computação em Nuvem**. São Paulo. M. Books do Brasil, 2013.

CHEMIN, Beatris Francisca. **Manual da Univates para trabalhos acadêmicos: planejamento, elaboração e apresentação como redigir e apresentar um trabalho científico**. Lajeado, Univates, 2015

COULOURIS, George. DOLLIMORE, Jean. KINDBERG, Tim. BLAIR, Gordon. **Distributed Systems – Concepts and Design, 5<sup>th</sup> Edition**. Pearson Education. Reino Unido, 2012.

COMER, E. Douglas. **Redes de Computadores e Internet**. Bookman. Porto Alegre, 2007.

ECMA, International. **ECMAScript Language Specification 7<sup>th</sup> Edition**. 2016. Disponível em: < <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf> > Acesso em: 03 mai. 2017.

FERNANDES, Jorge H. C. **O que é um Programa (Software)?**. 2017. Disponível em: <<http://www.cic.unb.br/~jhcf/MyBooks/iess/Software/oqueehsoftware.html>> Acesso em: 30 jul. 2017.

GERHARDT, Tatiana E; SILVEIRA, Denise T. **Métodos de pesquisa**. Porto Alegre, UFRGS, 2009. Disponível em: < <http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf> >. Acesso em: 1 Ago. 2017.

GLOBAL SIGN, **The Dangers of Self-Signed SSL Certificates**. Disponível em: <<https://www.globalsign.com/en/ssl-information-center/dangers-self-signed-certificates/>>. Acesso em: 5 Ago. 2017.

HASSELL, Jonathan. **Por que os profissionais de TI devem se preocupar desde já com a Internet das Coisas**. 2017. Disponível em: <<http://cio.com.br/tecnologia/2017/02/18/por-que-os-profissionais-de-ti-devem-se-preocupar-desde-ja-com-a-internet-das-coisas/>> Acesso em: 14 abr. 2017.

HIVEMQ, Dc Square GmbH. **How to Get Started with the lightweight IoT Protocol**. 2015. Disponível em: <<http://www.hivemq.com/blog/how-to-get-started-with-mqtt>> Acesso em: 28 abr. 2017.

IDC, International Data Corporation. **Internet of Things Spending Forecast to Grow 17.9% in 2016 Led by Manufacturing, Transportation, and Utilities Investments, According to New IDC Spending Guide**. 2017. Disponível em: <<http://www.idc.com/getdoc.jsp?containerId=prUS42209117>> Acesso em: 20 mar. 2017.



IEEE, 802.11. **Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) Specifications.** 2012. Disponível em:<<http://standards.ieee.org/getieee802/download/802.11-2012.pdf>> Acesso em: 08 mai. 2017.

IEEE, 802.11ac. **Wireless LAN Medium Access Control(MAC) and Physical Layer (PHY) Specifications. Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz.** 2013. Disponível em:<<http://standards.ieee.org/getieee802/download/802.11ac-2013.pdf>> Acesso em: 08 mai. 2017.

IoT. **Internet of Things: Vision, applications and research challenges.** 2012. Disponível em:<<https://irinsubria.uninsubria.it/retrieve/handle/11383/1762288/2389/IOT.pdf>> Acesso em: 25 fev. 2017.

JUNIOR, Celso Ferrarezi. **Guia do Trabalho Científico: do projeto à redação final.** São Paulo, Editora Contexto, 2011.

KABIR, Mohammed J. **Apache Server 2 A Bíblia.** Rio de Janeiro, 2002.

KINGSTON. **SDHC/SDXC UHS-I Classe 10 - 16GB-512 GB | Kingston.** 2017. Disponível em: <[https://www.kingston.com/br/flash/sd\\_cards/sda10](https://www.kingston.com/br/flash/sd_cards/sda10)> Acesso em: 27 fev. 2017.

LARMAN, Craig. **Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos.** Porto Alegre, Bookman, 2000.

MACHADO, Abimael. **Linguagem Java.** Disponível em: <<http://knoaam.com.br/linguagem-java-parte/>> Acesso em: 07 Ago. 2017.

MENDES, Douglas R. **Programação Java com ênfase em Orientação a Objetos.** São Paulo. Novatec, 2009.

MENDES, Douglas R. **Redes de Computadores – Teoria e Prática.** São Paulo. Novatec, 2016.

MILANI, André. **PostgreSQL: Guia do programador.** São Paulo. Novatec, 2008.

MIORANDI, Daniele. **Internet of things: Vision, applications and research challenges.** 2012. Disponível em: <  
<https://irinsubria.uninsubria.it/retrieve/handle/11383/1762288/2389/IOT.pdf> > Acesso em: 22 mar. 2017.

MONTEIRO, Emiliano S. **Certificados Digitais - Conceitos e Práticas.** Rio de Janeiro. Brasport, 2007.

MORIMOTO, E. Carlos. **Hardware: O Guia definitivo II.** Porto Alegre. Sul Editores, 2010.

MURATORI, J. R. e Dal Bó, P. H. **Automação Residencial Conceitos e Aplicação.** São Paulo. Edurece, 2013.

NIC. National Intelligence Council. **Six Technologies With Potential Impacts on US Interests Out to 2025.** 2008. Disponível em:<<https://fas.org/irp/nic/disruptive.pdf>> Acesso em: 20 mar. 2017.

NODE MCU. **Features of Node.** 2017. Disponível em: <  
[http://nodemcu.com/index\\_en.html#fr\\_54747361d775ef1a3600000f](http://nodemcu.com/index_en.html#fr_54747361d775ef1a3600000f)> Acesso em: 09 mai. 2017.

PI4J, Project. **About PI4J.** 2017. Disponível em: <<http://pi4j.com>> Acesso em: 28/04/2017.

PINHEIRO, José M. S. **Prova de Conceito no Projeto de Redes de Computadores.** 2010. Disponível em:  
 <[http://www.projetoderedes.com.br/artigos/artigo\\_prova\\_de\\_conceito\\_no\\_projeto\\_de\\_redes.php](http://www.projetoderedes.com.br/artigos/artigo_prova_de_conceito_no_projeto_de_redes.php)> Acesso em: 02/10/2017.

POSTGRESQL. **PostgreSQL 9.6 Documentation.** 2017. Disponível em: <  
<https://www.postgresql.org/docs/current/static/index.html>> Acesso em: 27/04/2017.

PRESSER, Mirko. **Internet of Things – Special Edition.** Europa. Autumn, 2011.

PRESSMAN, ROGER S. **Engenharia de Software 7ª edição.** São Paulo, Editora McGrawHill, 2011.

Raspberry Pi. **Ultimate guide to Raspberry Pi**. 2015. Disponível em:<  
<http://micklord.com/foru/Raspberry%20Pi%20Pages%20from%20Computer%20Shopper%202015-02.pdf>> Acesso em: 27 fev. 2017.

RICHARDSON, Matt. WALLACE, Shawn. **Primeiros Passos com o Raspberry Pi**. São Paulo. Novatec, 2013.

SERSON, Roberto R. **Programação orientada a objetos com Java 6**. Rio de Janeiro. Brasport, 2007.

SILVA, Bruna R. S. **Sistema de automação residencial de baixo custo para redes sem fio**. Porto Alegre, 2014. Disponível em:<  
<http://www.lume.ufrgs.br/bitstream/handle/10183/101188/000931903.pdf>> Acesso em: 30 mar. 2017.

SILVA, Maurício S. **Ajax com JQuery: Requisições Ajax com a simplicidade do JQuery**. Novatec. São Paulo, 2009.

SILVEIRA, Paulo, SILVEIRA, Guilherme, LOPES, Sérgio e et. Al. **Introdução à Arquitetura e Design de Software: Uma visão sobre a plataforma Java**. São Paulo. Casa do Código, 2011.

STALLINGS, William **Arquitetura e Organização de Computadores: Projeto para o Desempenho**. Prentice Hall, São Paulo, 2002.

TANENBAUM, S. Andrew e STEEN, Maarten V.. **Distributed Systems Principles and Paradigms 2nd Edition**. Prentice Hall, Amsterdam, 2007.

TANENBAUM, S. Andrew. WETHERALL, David. **Computer Networks 5th Edition**. Pearson, Amsterdam, 2011.

W3C, World Wide Web Consortium. **About W3C**. 2017. Disponível em:<  
<https://www.w3.org>> Acesso em: 10 abr. 2017.



**UNIVATES**

R. Avelino Tallini, 171 | Bairro Universitário | Lajeado | RS | Brasil  
CEP 95900.000 | Cx. Postal 155 | Fone: (51) 3714.7000  
[www.univates.br](http://www.univates.br) | 0800 7 07 08 09